אקדמון
בית ההוצאה של הסתדרות הסטודנטים של האוניברסיטה העברית

דוד סלומון

שפת התכנות FORTRAN IV

מהדורה שניה

ירושלים תשל"ג

דוד סלומון

שפת התכנות FORTRAN IV

מהדורה שניה

אקדמון

בית ההוצאה של הסתדרות הסטודנטים של האוניברסיטה העברית

ירושלים, תשל"ב - 1971

צולם ונדפס באופסט האמנים, ברנשטיין את חן, ירושלים

מ ב ו א

חוברת זו נוצרה כתוצאה מסדרת הרצאות שנתנו בשנים תשכ"ט-תש"ל באוניברסיטה העברית לתלמידים מכל הפקולטאות.

החמר המובא כאן מהוה מבוא לשפת ה- FORTRAN , וכולל את כל הדברים הבסיסיים המאפשרים כתיבת תכנית.

על מנת לשמור על כלליותה של החוברת, לא הוכנסו לתוכה, במידת האפשר, תכונות האופיניות למחשב מסוים.

ברוב המקרים יספיק החמר הכלול כאן לעבודה שוטפת ובמקרה של בעיות מורכבות יוכל הקורא לפנות לספרות המומלצת כאן.

תודתי העמוקה נתונה למר מ. לב על עזרתו הרבה בכל שלבי כתיבת החוברת.

ספרות מומלצת:

1) Daniel D. McCracken: Introduction to FORTRAN IV Programming. J. Wiley & Sons.

2) FORTRAN II and IV for the IBM 360. Form No. 36-0854.

3) FORTRAN IV for the CDC 6400. Publication No. 60174900.

תוכן

עמוד

פרק 1

קבועים ומשתנים, מבנה ניופס קידוד

מחשב הוא מכונת חישוב ואינו מכונת חשיבה. הוא מסוגל לחשב (במהירות ובדיקנות) אולם אינו מסוגל לחשוב. לכן כל המחשבה חיבת להעשות ע"י האדם המשתמש במחשב. כדי לפתור בעיה בעזרת מחשב לא מספיק להסביר לו מהי הבעיה ולתת לו את הנתונים. הוא אינו מסוגל לחשוב ולכן לא ימצא דרך לפתרון הבעיה. כדי לפתור בעיה בעזרת המחשב יש קודם כל לפתור אותה לבד. משפט זה נשמע כאילו הוא מכיל בתוכו סתירה אולם הכונה היא שיש לפתור את הבעיה רק באופן עקרוני, כלומר למצוא נוסחה שתוביל לפתרון. נוסחה זו אפשר להגיש למחשב והוא ימצא את הערך המספרי שלה (כלומר יחשב את המספר, או המספרים שהם הפתרון המבוקש).

לדוגמא, יש לפתור את המשואה הרבועית $2x^2 - 3x + 4 = 0$. על מנת לבצע זאת בעזרת המחשב יש קודם כל לפתור באופן כללי את בעית המשואה הרבועית $ax^2 + bx + c = o$, לקבל את הבטוי $x = (-b \pm \sqrt{b^2 - 4ac}) / 2a$ ולהגיש בטוי זה למחשב, המחשב יכול לחשב את ערכיו של הבטוי לגבי $a = 2 \quad b = -3 \quad c = 4$ ולקבל את שני הפתרונות המבוקשים.

כל מה שהמחשב עשה היה רק חישוב של נוסחה, כלומר ביצוע פעולות אריתמטיות על נתונים מספריים וקבלת תוצאה סופית שהיא מספר. המחשב לא הבין שהוא פתר משואה רבועית ולמעשה הוא גם לא פתר אותה. כל מה שהוא עשה היה חישוב ערכה של נוסחה ע"י ביצוע פעולות אריתמטיות.

כדי לפתור בעיה בעזרת מחשב יש לכן לבצע ארבעה שלבים.

1) הבנת הבעיה. מחשב לא יוכל לפתור בעיה שהתכניתן עצמו לא הבין אותה.

2) התוית הצעדים לפתרון. יש לערוך רשימה של צעדים שביצועם יוביל לפתרון.

3) רישום הבטויים המתמטיים אותם יש לחשב בצעדים השונים.

4) כתיבת אותם בטויים למחשב בצורה אותה הוא יכול לקרוא ולהבין.

ביצוע השלב הרביעי הוא למעשה כתיבת תכנית למחשב. התכנית היא אוסף של פעולות אלמנטריות המבוצעות ע"י המחשב אחת אחת ושבסופן מתקבלת התוצאה המבוקשת. כדי שהמחשב יוכל לבצע את הפעולות יש להגיש לו אותן בשפה אותה הוא יכול להבין. לשם כך יש לכל מחשב שפה (או שפות) אותה הוא יכול להבין. כאן תתואר השפה הנקראת FORTRAN (לכבוד המליפ Formula Translation) שהיא שמושית מאוד ומובנת ע"י רוב המחשבים הקימים כיום.

שפה זו נוצרה בשנים 1956-57 ועברה מאז כמה שלבי התפתחות. באופן עקרוני לא צריכים להיות הבדלים בשפה בין מחשב למחשב ותכנית הכתובה FORTRAN צריכה לעבוד על כל המחשבים המבינים שפה זו. למעשה יוצרים ההבדלים במבנה בין מחשב למחשב, הבדלים קטנים ומגבלות קטנות בשפת ה- FORTRAN בכל מחשב. לכן תכנית FORTRAN שעובדת על מחשב אחד, תדרוש אולי כמה שנויים כאשר ירצו להפעילה על מחשב אחר.

כאן תתואר השפה מתוך זיקה למחשב המורכב באוניברסיטה העברית (מסוג CDC 6400) ולכן יופיעו פה ושם הערות המתארות מגבלות או תכונות של השפה שהן אופיניות רק למחשב זה.

כמו בכל שפה קימות גם ב- FORTRAN אותיות (הנקראות סמלים יסודיים או קרקטרים יסודיים), קימות מלים המורכבות מהאותיות וקימים חקי דקדוק בעזרתם מרכיבים משפטים מהמלים. הקטע המינימלי בתכנית שהוא בעל "מובן" הוא משפט (Statement). כדי להבהיר את משמעות המלה "מובן" יש להשוות את המשפטים בשפת ה- FORTRAN למשפטים בשפה המתמטית הרגילה.

משפט מתמטי רגיל הוא סטטי. הוא מציין עובדה אולם אינו דורש לבצע כל פעולה. לדוגמא: הזהות $(a+b)(a-b) = a^2-b^2$ מתארת עובדה הנכונה לכל שני מספרים a b , אולם זהות זו אינה דורשת לחשב את אחד האגפים או כל דבר אחר.

במתמטיקה אפשר לומר שאם קים $ax+b = o$ אז קים $x = -\frac{b}{a}$. זוהי דרך לפתרון המשואה אולם כאן אין כל פקודה האומרת לפתור את המשואה. זהו רק ציון עובדה.

בשפת ה- FORTRAN , לעומת זאת, מהוים המשפטים פקודות שיש לבצען. כל משפט (פרט ליוצאים מן הכלל בודדים) מכיל בתוכו פעולה (action) שיש לבצע אותה. כאן אין משמעות למשפט כמו $(a + b)(a - b) = a^2 - b^2$ אלא רק למשפטים מהצורה: חשב את $(a + b)(a - b)$

חשב את $a^2 - b^2$ וכו'.

משפט בשפת ה- FORTRAN יכול להיות לדוגמא:

<u>אם</u> $a x + b = o$ <u>אז</u> עשה כך וכך....

בשפה המתמטית הרגילה נוהגים לפעמים לכתוב בטויים שאינם חד-משמעיים, מתוך כונה שהקורא יבין את המשמעות מתוך ההקשר בו נכתבו הדברים. לבטוי המתמטי $\ln(m^2 + g^2)$ יכול להיות המובן של $l \cdot n \cdot (m^2 + g^2)$ ויכול להיות המובן של

הלוגריתמוס הטבעי של $(m^2 + g^2)$. בשפת ה- FORTRAN חיבים המשפטים להיות חד-מַשמעיים. המחשב לא יכול להבין משפט מתוך ההקשר בו הוא מופיע, מכיון שהוא אינו מסוגל להבין מאומה. לכן, משפט בעל "מובן" בשפת ה- FORTRAN הוא משפט הכתוב לפי כללי הדקדוק של השפה (ולכן הוא חד-משמעי) ואומר למחשב לבצע איזשהי פעולה.

הסמלים היסודיים בשפת ה- FORTRAN מתחלקים לשלוש קבוצות.

1) האותיות הלטיניות הגדולות (Capital Letters) A - Z.

2) הספרות 0-9.

3) הסמנים המיוחדים $. , b = ( ) / * - +

על מנת להבחין בין הספרה אפס לבין האות O, נכתבת האחרונה בצורה Ø .

הסימן b בא לציין את הסימן הריק (blank). הסמן הריק הוא אות חוקית בשפת ה- FORTRAN ופרושה, להשאיר מקום ריק על גבי כרטיס הניקוב או על גבי ניר ההדפסה. על מנת למנוע בלבול נרשמה כאן האות b בכל פעם שצריך היה להשאיר מקום ריק. האות b עצמה איננה אות גדולה ולכן אינה סמל יסודי בשפה.

המלים הראשונות שנכיר בשפה הן שמות של קבועים ושל משתנים. כדי להבין את המשמעות של "קבועים" ושל "משתנים" נתבונן בבטוי $b^2 - 4ac$. בטוי זה מכיל את המספרים 2,4 ואת האותיות a, b, c . כל אחת מהאותיות מיצגת מספר שערכו עלול להיות שונה בכל פעם.

ה"קבועים" בשפת ה- FORTRAN נותנים אפשרות להכניס מספרים לתוך המשפטים של השפה. ה"משתנים" שבשפה נותנים אפשרות להכניס אותיות המיצגות מספרים, ממש כמו בבטויים האלגבריים הרגילים.

הקבועים מתחלקים לשני סוגים. שלמים וממשיים.

<u>הגדרה:</u> קבוע שלם הוא סדרה של אחת עד שמונה-עשרה ספרות עשרוניות, שלפניה יכול לבוא סמן.

<u>דוגמאות:</u> 384155537 + 456 23 - 1. כאשר הקבוע הוא חיובי אפשר להשמיט את סמן ה + .

<u>הגדרה:</u> קבוע ממשי הוא סדרה של אחת עד ארבע-עשרה ספרות עשרוניות, המכילה נקודה עשרונית או מעריך עשרוני או שניהם, ולפניה יכול לבוא סמן.

הצורות הכלליות של קבוע ממשי יכולות להיות

n.n n. .n n.nE$\pm$s n. E$\pm$s .nE$\pm$s nE$\pm$s

<u>דוגמאות:</u> 3.1415926 1. .015

0.1 E 32 -7.E 8 -.9 E - 17 57 E - 82

האות E (מהמלה Exponent) פרושה שיש לכפול את המספר בחזקה שלמה (חיובית או שלילית) של עשר.

לדוגמא: -9. E - 17 פרושו המספר הממשי $-9 \cdot 10^{-17}$.

המשתנים מתחלקים גם הם לשני סוגים, - שלמים וממשיים. מאחר וגודל משתנה מכיל בכל פעם מספר אחר, הרי שאי אפשר לתת למשתנה שם שיהיה מורכב מהמספר אותו הוא מכיל, (כפי שזה נעשה בגדלים קבועים) אלא יש להמציא שם המורכב מקרקטרים של FORTRAN . שם של משתנה חיב להתאים לכללים הבאים:

1. שם של משתנה מכיל בין אחד לשבעה קרקטרים.

2. הקרקטרים המותרים הם רק אותיות וספרות ולא סימנים מיוחדים.

3. הקרקטר הראשון בשם הוא אות.

4. שם של משתנה שלם חיב להתחיל באחת מהאותיות הבאות: I J K L M N . שם של משתנה ממשי חיב להתחיל באות אחרת.

הערות:

1) לכלל מס' 4 קים יוצא מן הכלל שיתואר בעמוד 25.

2) ארכו המקסימלי של השם יכול להיות שונה משבעה קרקטרים, במחשבים אחרים.

דוגמאות:

I   GRØSS   NET   KLM   ØK   FØRTRAN

MAX40   I032J   CØNSTAT   GAMAO   XY

את המשפטים בשפת ה- FORTRAN רושמים על טופס מיוחד הנקרא טופס קידוד (Coding Sheet). כל משפט נרשם בשורה (או בכמה שורות), ואח"כ מנקבים את תכנו של הטופס על כרטיסי ניקוב. כל שורה בטופס מתנקבת על כרטיס. מאחר וכרטיס ניקוב מכיל 80 מקומות לניקוב הרי שכל שורה בטופס חיבת להכיל 80 מקומות (80 משבצות). בזמן כתיבת התכנית יש להקפיד שכל קרקטר יכנס בדיוק למשבצת אחת. הקרקטר b נרשם ע"י השארת משבצת אחת ריקה. כל שורה בטופס מחולקת לארבעה שדות.

1) מקומות 1-5. משמשים כדי למספר את המשפטים במספרי זהוי (Statement Number).

2) מקום 6. משמש לסמון המשכיות (Continuation).

3) מקומות 7-72. משמשים לרישום המשפט עצמו.

4) מקומות 73-80. משמשים למיספור שוטף של כרטיסי התכנית (Sequencing).

תאור השדות:

1) כל משפט בשפה יכול (אולם בדרך כלל אינו חייב) להיות מזוהה ע"י מספר. מספר זה מאפשר לפנות אל המשפט ממקומות אחרים בתכנית. מספרי הזהוי מורכבים מאחת עד חמש ספרות והם אינם חייבים להופיע בסדר שוטף או בכל סדר שהוא. אסור ששני משפטים יזוהו ע"י אותו מספר זהוי.

2) כאשר משפט מסוים ארוך ואינו נכנס לשורה אחת על הטופס, אפשר להמשיכו בשורה הבאה. למעשה יכול כל משפט להשתרע על עשר שורות (שורה מקורית ומקסימום תשע שורות המשך). על שורות ההמשך יש לציין שהן מהוות המשך של השורה הקודמת. לשם כך כותבים במקום השישי בשורה, קרקטר כלשהו (פרט לקרקטרים b o). קרקטר כלשהו המופיע במקום השישי בשורה, מראה שהשורה היא המשך של השורה הקודמת.

3) המשפט עצמו נרשם במקומות 7-72. אם הוא ארוך מדי, אפשר להמשיכו החל ממקום 7 בשורה הבאה, אולם יש לסמן המשכיות ע"י כתיבת קרקטר כלשהו במקום 6 בשורת ההמשך. סמן ההמשכיות מופיע רק בשורות ההמשך ולא בשורה המקורית.

4) מקומות 73-80 אינם חלק מהתכנית והמחשב כלל אינו מתיחס אליהם. הם משמשים למיספור שוטף של השורה. מיספור זה חשוב משתי סיבות:

א. הטופס מנוקב על חבילת כרטיסים ואם רוצים למצוא כרטיס מסוים בחבילה זו, הנוח ביותר הוא למצאו בעזרת המיספור השוטף שלו.

ב. אם הכרטיסים בחבילה מתפזרים, הרי קל לסדר אותם שוב ע"י מיונם (במכונת מיון) לפי המיספור השוטף.

כללי הכתיבה על טופס הקידוד הם פשוטים. קיים למעשה רק כלל יחיד האומר שהכתיבה על הטופס היא חפשית פרט לכך שאסור לחרוג מתחומי השדות שצוינו.

כתיבה חפשית פרושה שאין צורך להתחיל לכתוב מתחילת השדה, אפשר להשאיר רוחים באופן חפשי בין מלה למלה ואפילו באמצע מלה, אפשר להדביק שתי מלים ואפילו את כל המלים במשפט.

דוגמאות:

1) אם מספר הזהוי של משפט כלשהו הוא 105 אזי מספר זה יכול להרשם בכל מקום שהוא בין המקומות 5-1 בשורה.

כל הצורות הבאות 1bb 05 1b0b5 bb105 105bb הן חוקיות.

2) המשפט 14 GØ TØ הוא משפט בשפת ה- FORTRAN ואפשר לרשום אותו בצורות הבאות:

GbØT bØ14 GØTØ14 GØTØb14 שכולן חוקיות.

<u>כרטיס הערה (Comment)</u>.

אפשר , בכל מקום בתכנית, לרשום שורה שתכיל רק הערה ולא משפט ב- FØRTRAN . הערות כאלה חשוב לכתוב כדי להבהיר את מבנה התכנית לכל אדם הקורא אותה. כדי להודיע

למחשב ששורה מסוימת מכילה הערה ולא משפט, רושמים את האות C בעמודה 1. כאשר המחשב פוגש את האות C בעמודה 1 הוא "מבין" ששורה זו אינה מכילה משפט ואינו מתיחס אליה. גם בשורה כזו מותר לרשום רק קרקטרים של FORTRAN ולא שום סמל אחר.

פרק 2

פעולות אריתמטיות, בטויים אריתמטיים ומשפטי יחוס

הקבועים והמשתנים נוצרו כדי לאפשר כתיבת נוסחאות מתמטיות בשפת FORTRAN בצורה שתהיה דומה ככל האפשר לכתיב המתמטי המקובל. לשם כך יש ליצור בשפה גם פעולות אריתמטיות, וקימות חמש כאלה.

חבור +, חסור - , כפל * , חילוק / והעלאה בחזקה **, יש להבין שסימן ההעלאה בחזקה הוא סימן יחיד המורכב משני קרקטרים. אין חשש לבלבול בינו לבין שני סמני כפל מאחר ושני סמני כפל אינם מופיעים לעולם ברציפות אחד אחרי השני.

רק חמש פעולות אלה קימות בשפה, וכל חישוב יש לפרק לשורה של פעולות אלה.

<u>הגדרה:</u> בטוי אריתמטי הוא צרוף של קבועים, משתנים, פעולות חשבון וסוגריים, שיש לו ערך מספרי יחיד. הבטוי יכול להכיל גם פניות לפונקציות (ראה עמוד 68).

על מנת שלבטוי יהיה ערך מספרי הוא חייב לקיים את הכללים הבאים:

1) שתי פעולות אריתמטיות לא תבואנה אחת אחרי השניה.

2) בטויים בעלי עדיפות בחישוב חייבים להכנס לסוגרים.

3) ללא סוגרים קיים סדר עדיפויות בביצוע הפעולות. קודם תבוצענה ההעלאות בחזקה, אח"כ פעולות הכפל והחילוק ולבסוף פעולות החיבור והחיסור.

4) שורה של פעולות כפל וחילוק ללא סוגריים תחושב משמאל לימין. כלומר A/B*C יחושב בתור (A/B)*C ולא בתור A/(B*C) .

5) אם הבטוי האריתמטי מכיל רק גדלים שלמים יהיה ערכו שלם, אם הוא מכיל רק גדלים ממשיים או אם הוא מכיל גדלים שלמים וממשיים, יהיה ערכו ממשי.

המקרה היחיד בו אסור לערב גודל ממשי וגודל שלם הוא העלאת גודל שלם בחזקה ממשית. כלומר 3**2.5 הוא בטוי אסור ובמקומו יש לכתוב 3.**2.5.

דוגמאות: הנוסחאות המתמטיות הבאות, נכתבות בתור בטויים אריתמטיים:

$a\ b$	A*B	
$a\ (-b)$	A*(-B)	או -A*B
$a + 2$	A + 2	או A+2.
$-(a + b)$	-(A + B)	או - A - B
a^{i+2}	A**(I + 2)	
$a^{(\frac{b}{2} + c)}$	A**(B/2 + c) ?	
$a\ b/c\ d$	A*B/(C*D)	
$a\ (x+b\ (x+c))$	A*(X+B*(X+C))	
$a/(1 + \frac{b}{3.1 + c})$	A/(1 + B/(3.1 + C))	

הערה: הבטוי האריתמטי I / J מכיל רק גדלים שלמים ולכן התוצאה שלו היא מספר שלם. במקרה והתוצאה היא בכל זאת לא שלמה (כמו במקרה 3/2) יקצץ המחשב את התוצאה עד למספר השלם הקרוב ביותר. לכן 3/2 נותן 1 בתור תוצאה. עובדה זו עלולה להיות חשובה כפי שמראה הדוגמא הבאה:

$4 * 3/2 = 12/2 = 6$ ואילו $3/2 * 4 = 1 * 4 = 4$

כלומר הערך של בטוי מהצורה I*J / K שונה מערכו של הבטוי J/K*I .

תרגיל: הפוך את הנוסחאות הבאות לבטויים אריתמטיים:

$(X + Y)^3$ $a + \frac{b}{c + d^2}$ $\sqrt{j}$ $1/(1 - \frac{v^2}{c^2})f$-

תרגיל: כל אחת מהנוסחאות שלמטה תורגמה לבטוי אריתמטי הכולל שגיאה (או שגיאות). מהן?

$\frac{a\ b}{c+2} \longrightarrow$ AB (C+2)

$\frac{1}{a^2}\ (\frac{f}{8})^a \longrightarrow$ 1/A**2 (F/8)** A

בטוי אריתמטי יכול להיות מסובך ביותר, בכל זאת הוא אינו מהווה משפט שלם ב- FORTRAN . הסיבה לכך היא שבטוי אריתמטי מכיל נוסחה, אולם אינו מכיל הוראה למחשב לחשב את הנוסחה ואינו אומר למחשב מה לעשות עם המספר שהוא תוצאת חישוב הנוסחה. כדי להשלים בטוי אריתמטי למשפט שלם יש לכן להוסיף שני דברים.

א. להורות למחשב לחשב את הבטוי.

ב. לומר למחשב מה לעשות עם התוצאה.

דברים אלה נעשים ע"י מש׳ יחוס אריתמטי. צורתו הכללית של המשפט היא a = e כאשר a הוא שם של משתנה, e הוא בטוי אריתמטי.

המשמעות - זו פקודה למחשב האומרת לו לחשב את הבטוי האריתמטי e ולהעביר את המספר שהוא תוצאת החישוב, לתוך המשתנה שבאגף שמאל.

לאחר בצוע המשפט יהיה למשתנה a הערך של הבטוי האריתמטי, וכל ערך קודם שהיה לו, ילך לאיבוד.

דוגמאות:

1) $I = 1$ אגף ימין הוא הבטוי האריתמטי הפשוט 1. המחשב מכניס את ערכו של אגף ימין (המספר 1), לתוך המשתנה השלם I. התוכן הקודם של I נעלם ומעתה יכיל I את המספר 1.

2) $I = J$ כאן ערכו של אגף ימין הוא המספר השלם הנמצא בתוך המשתנה J. מספר זה עובר גם לתוך המשתנה I והוא יהיה להבא ערכו של המשתנה I (עד השנוי הבא של I). ערכו של J אינו משתנה.

על מנת שלמשפט האחרון תהיה משמעות חייב J להיות מוגדר (כלומר להכיל מספר שלם כלשהו). אם J אינו מוגדר, המחשב יודיע על שגיאה ויפסיק את בצוע התכנית.[1]

3) $I = J + 1$ כאן ערכו של אגף ימין גדול באחד מהערך של J. לכן אם J הכיל 2, יכנס לתוך I המספר 3.

4) $I = I + 1$ לתוך I יכנס הערך שהיה לו מקודם פלוס 1. לכן אם I הכיל מקודם 5 יהיה בו כעת 6.

הדוגמא האחרונה מראה את ההבדל בין סמן השויון ב-FORTRAN לבין אותו סמן במתמטיקה.

במתמטיקה יש לסמן השויון משמעות סטטית. הוא מודיע על קיומו של שויון ותו לא. כאשר כותבים $a = b$ מתכונים לכך ש- a b שוים לא רק בגודלם אלא גם באופים. שניהם מספרים או שניהם וקטורים וכו'. אם קיים $a = b$ אז גם $b = a$ אם $a + b = x + c$ אז מהנחת השויון אפשר למצוא את x.

---

1) קימים מחשבים שאינם מפסיקים את העבודה במקרה כזה.

לעומת זאת ב- FORTRAN יש לסמן השויון משמעות דינמית. הוא מהוה אופרטור הגורם למחשב לבצע פעולה מסוימת ואינו מראה על קיומו של איזהו שויון. סימן השויון הוא הוראה למחשב לחשב את אגף ימין ולהכניס את ערכו לאגף שמאל. אגף ימין הוא בטוי אריתמטי ואגף שמאל הוא משתנה. לכן אין לומר שהם שוים. כמו כן I = 1 הוא משפט חוקי ואילו 1 = I הוא משפט שאינו קיים בשפה. המשפט A = X + 1 אינו מאפשר לנו למצוא את ערכו של X אלא אומר שערכו של X יוגדל ב-1 והמספר שיתקבל יוכנס למשתנה A .

במשפט יחוס יכולים שני האגפים להיות שונים באופים. אחד יכול להיות ממשי והשני שלם.

במקרה I = X (אגף ימין ממשי ואגף שמאל שלם) יקוצץ ערכו של אגף ימין עד לשלם הקרוב ביותר ושלם זה יוכנס לאגף שמאל. במקרה X = I יהפוך ערכו של אגף ימין לממשי ומספר ממשי זה יוכנס לאגף שמאל.

דוגמאות: I = 3.5 לתוך I יוכנס 3.
X = 5 לתוך X יוכנס 5. .

דוגמאות: המשואות המתמטיות הבאות

$$\pi = 3.14 \qquad y = (10^{-6} + a \times 4)^{2/3} \qquad \beta = \frac{-1}{2x} + \frac{a^2}{4x^2}$$

נכתבות בתור משפטי יחוס

```
BETA = - 1./(2*X) + A** 2/(4*X*X)
Y = (1. E - 6 + A*X** 4)** .66667      PI = 3.14
```

תרגיל: מה השגיאות במשפטי היחוס הבאים?

$-V = A + B$ $A * X = B * X$ $Z = 1. / - 2. * A$

תרגיל: חשב את ערכו של אגף ימין ומה נכנס לאגף שמאל בכל אחד מהמקרים:

$A = 2. * 6./4.$ $I = 2 * 10/4$ $A = 2. * (10./4.)$

$A = 6.0 * (1./6.0)$ $A = 19/4 + 5/4$ $I = 100 * (99/100)$

לסיכום: משפט יחוס אריתמטי משמש כדי להכניס נתונים מספריים לתוך משתנים. צורתו a = e כאשר e- בטוי אריתמטי, a- שם של משתנה.

זהו המשפט הראשון שאנו מכירים בשפה והוא גם אחד המשפטים השמושיים ביותר.

$A = 2/3 = 0$

$I = 2*(10/4) = 4$ (במקום 5)

$A = 2*(10/4) = 4$

$A = 2.*(10./4.) = 5.$

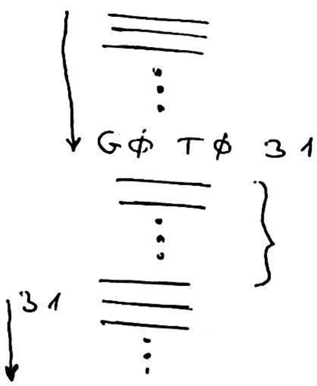
פרק 3

קפיצות בתכנית

המחשב מבצע את התכנית, פקודה אחר פקודה לפי הסדר בו מופיעות הפקודות, כלומר לפי הסדר בו נרשמו הפקודות על הטופס. בהרבה מקרים מעונינים לדלג על פקודות מסוימות או מעונינים לחזור ולבצע פקודות מסוימות פעם נוספת. לכן יש לספק פקודות (או משפטים) מיוחדות שתאפשרנה לחזור בחזרה או לקפוץ קדימה ולבצע פקודות שלא לפי הסדר בו הן מופיעות.

הפקודה הראשונה מסוג זה נקראת "GØ TØ ללא תנאי". צורתה הכללית GØ TØ n כאשר n הוא מספר זהוי של פקודה כלשהי בתכנית. המשמעות - במקום לבצע את הפקודה שאחרי ה- GØ TØ, קופץ המחשב לפקודה שמספרה n, מבצע אותה וממשיך לבצע ממנה.

מקרה זה מתואר סכמטית בציור. החצים מראים את סדר בצוע הפקודות. הפקודות באזור המסומן, לא תבוצענה.



כאשר כותבים GØ TØ 31 חיבים לדאוג לכך שתימצא פקודה שמספרה 31 ורק פקודה יחידה כזו.

הפקודה השניה מסוג זה נקראת "GØ TØ מחושב".

צורתה הכללית: $G\emptyset\ T\emptyset\ (n_1, n_2, \ldots, n_k), i$

כאשר $n_1\ n_2 ... n_k$ הם מספרי זהוי של פקודות. i- שם של משתנה שלם כלשהו.

המשמעות – אם המשתנה השלם i מכיל את המספר 3. יקפוץ המחשב לפקודה שמספרה n_3 . באופן כללי – אם i מכיל את המספר L יקפוץ המחשב לפקודה שמספרה n_L . לדוגמא –

M = 5

GØ TØ (279, 1, 3, 1002, 1, 17), M

המחשב יקפוץ לפקודה מספר 1 (הפקודה שמספרה הוא חמישי ברשימה).

יש לדאוג לכך שהמשתנה השלם יכיל מספר הנמצא בתחום הנכון. בדוגמא שלמעלה יכול M להכיל רק מספר בין אחד לשש. במקרה הכללי יכול i להכיל רק מספר בין 1 ל k . ערכו של k אינו מוגבל.

שתי פקודות אלה מהוות משפטים בשפה מכיון שהן פקודות שהמחשב "מבין" ומבצע.

הדוגמא הבאה היא תכנית שלמה ב- FORTRAN . תכנית זו קוראת שלושה מספרים לתוך המשתנים N,A,B ומבצעת פעולות על A B לפי ערכו של N .

אם N = 0 מודפסים ערכי A B .

אם N = 1 מחושבים משתנים חדשים C D E לפי הנוסחאות $C = A + B \quad D = A^2 + B^2 \quad E = C/D$, ומודפסים ערכי C D E .

אם N = 2 , מחושבים C D E לפי הנוסחאות $C = A - B \quad D = A^2 - B^2 \quad E = C/D$ ומודפסים ערכי C D E .

```
   read  N, A, B
   N = N + 1
   GØ  TØ (10, 20, 30), N
10 print  A, B
   STØP   GØ TØ 40
20 C = A + B
   D = A** 2 + B** 2
   E = C/D
   print C, D, E
   GØ  TØ 40
30 C = A - B
   D = A** 2 - B** 2
   E = C/D
   print C, D, E
40 STØP
   E N D
```

<u>הערות:</u>

1) הפקודות read print נכתבו באותיות קטנות מאחר ועדיין לא תוארה צורתן הכללית (ראה פרק 7).

2) הפקודה STØP פרושה שנגמר בצועה של התכנית ועל המחשב לעצור.
פקודה זו יכולה להופיע בכל מקום בתכנית (לא דוקא בסופה) ויכולה להופיע כמה פעמים בתכנית. בדוגמא שלמעלה מופיעה הפקודה פעמיים וזה מכיון שהחישוב נפסק בנקודות שונות של התכנית בהתאם לערכו של N .

3) הפקודה END חיבת לבוא בסוף התכנית ומשמעותה היא - נגמרו כרטיסי התכנית והכרטיסים הבאים הם מסוג אחר, (כרטיסי נתונים או כרטיסים של תכנית משנה).

4) לתוך המשתנים - N A B נקראים ערכים מספריים ע"י הפקודה read . פקודה זו קוראת את המספרים מכרטיס מיוחד הנקרא כרטיס נתונים והמסופק ע"י התכניתן יחד עם התכנית.

5) הפקודה N = N + 1 גורמת לכך שערכו של N יגדל באחת ויהיה לכן בין 1 ל 3 . לכן N יכול לשמש בתור משתנה בפקודה GØ TØ (10, 20, 30), N ,

הפקודה השלישית הגורמת לקפיצות בתכנית נקראת " IF - אריתמטי".

צורתה $IF\ (e)\ n_1,\ n_2,\ n_3$

כאשר e - בטוי אריתמטי, n_j - מספרי זהוי של פקודות.

המשמעות - אם ערכו של הבטוי e הוא שלילי - קפוץ לפקודה שמספרה n_1 . אם הוא אפס - קפוץ לפקודה שמספרה n_2 ואם הוא חיובי - קפוץ לפקודה שמספרה n_3 .

דוגמא: IF (N - 1) 10, 20, 30

במקרה זה יקפוץ המחשב לפקודה 10 אם N - 1 שלילי כלומר אם N קטן מאחד. הוא יקפוץ לפקודה 20 אם N = 1 ולפקודה 30 אם N > 1 .

<u>הערה:</u> פקודה זו אפשר לרשום בדוגמא הקודמת במקום שתי הפקודות

N = N + 1

GØ TØ (10, 20, 30), N

פרק 4

בטויים לוגיים "IF - לוגי"

קיימים בטויים המטפלים במספרים (או בגדלים בעלי ערך מספרי) אולם להם עצמם אין ערך מספרי. בטויים כאלה יכולים להצהיר משהו לגבי ערכים מספריים, משהו היכול להיות "נכון" או "לא נכון".

לדוגמא: הבטוי $I > 3$ מטפל בגדלים המספריים 3, I אולם לו עצמו אין ערך מספרי. הוא טוען שהמשתנה I מכיל מספר הגדול משלוש. הבטוי יהיה "נכון" אם I באמת מכיל מספר גדול משלוש. הבטוי יהיה "לא נכון" בכל מקרה אחר.

בטויים מסוג זה יקראו בטויים לוגיים ונראה איך הם נכתבים בשפה.

ראשית מוגדרות שש פעולות יחוס. אלה הן הפעולות

.GT. .GE. .NE. .EQ. .LE. .LT.

הן משמשות להשואה בין שני גדלים מספריים (בין שני בטויים אריתמטיים) ומשמעותן -

.LT.	פרושו	קטן מ-	Less Than
.LE.	פרושו	קטן שוה	Less Equal
.EQ.	פרושו	שוה	Equal
.NE.	פרושו	לא שוה	Not Equal
.GE.	פרושו	גדול או שוה	Greater Equal
.GT.	פרושו	גדול מ-	Greater Than

ההשואה נעשית ע"י כתיבת פעולת יחוס בין שני בטויים אריתמטיים.

לדוגמא: X .EQ. Y . בטוי זה הוא <u>בטוי יחוס</u> והוא יהיה "נכון" אם המשתנה X מכיל מספר השוה למשתנה Y .

<u>דוגמא</u>: J + 5*K .GE. 37 - L

שני האגפים מהוים בטויים אריתמטיים. הבטוי כולו יהיה "נכון" אם אגף שמאל הוא גדול או שוה לאגף ימין.

על מנת לאפשר שימוש גמיש ויעיל בבטויי היחוס, הופכים אותם לבטויים לוגיים ע"י הכנסת שלושת הפעולות הלוגיות.

פעולות אלה הן .NØT. .ØR. .AND.

<u>לדוגמא</u>: הבטוי הלוגי A.LT.0 .ØR.B.LT.0 יהיה "נכון" אם A או B (או שניהם) יהיו שליליים.

הבטוי הלוגי I.GT.3.AND.I.LT.5 יהיה "נכון" אם I גדול משלוש וגם קטן מחמש.

הבטוי הלוגי .NØT.X.GE.10 יהיה "נכון" אם X קטן מ-10.

<u>הגדרה</u>: בטוי לוגי הוא צרוף של בטויי יחוס ופעולות לוגיות, צרוף שאין לו ערך מספרי אולם תמיד יש לו ערך "נכון" או "לא נכון" (True או False).

<u>הערות</u>:

1) שתי פעולות לוגיות לא תבאנה אחת אחרי השניה פרט לשני המקרים הבאים .AND..NØT. או .ØR. .NØT . (במקרים אלה באות שתי נקודות אחת אחרי השניה ברציפות).

2) בבטוי לוגי אפשר להשתמש בסוגריים אולם ללא סוגריים יבוצעו הפעולות בסדר העדיפויות הבא. תחילה יחושבו הבטויים

האריתמטיים, אח"כ בטויי היחוס, אח"כ .NØT. , אח"כ .AND. ולבסוף .ØR. .

3) A.ØR.B פרושו A או B או שניהם.

דוגמאות: 1) X.EQ.Y-1.ØR. X.EQ.Y+1

כאן מחושבים תחילה הבטויים האריתמטיים Y+1 Y-1 (שהם מספרים). אח"כ בטויי היחוס X.EQ.Y+1 X.EQ.Y-1 (שערכיהם "נכון" או "לא נכון") ולבסוף הפעולה .ØR. .

2) A.EQ.0.AND.B.NE.0.ØR.C.GT.3

כאן מחושבים בטויי היחוס, אח"כ הפעולה .AND. ולבסוף הפעולה .ØR. לכן ערכו של הבטוי הוא כאילו הוצבו הסוגריים הבאים (A.EQ.0.AND.B.NE.0).ØR.C.GT.3 .

חשוב לשים לב לנקודה זו מאחר והצבת הסוגריים בצורה שונה, למשל כמו בדוגמא A.EQ.0.AND.(B.NE.0.ØR.C.GT.3) משנה את ערכו של הבטוי.

<u>הערה:</u> הפעולות .ØR. .AND. מופיעות בין שני בטויי יחוס בצורה (...).AND.(...) ואילו .NØT. מופיעה משמאל לבטוי יחוס ופעולתה להפוך את משמעותו של הבטוי.(...).NØT. .

בטויים לוגיים משמשים במשפט אחד בשפה.

המשפט נקרא " IF - לוגי".

צורתו הכללית IF (1) S כאשר 1 - בטוי לוגי, S- משפט אחר ב- FORTRAN . המשפט S יכול להיות משפט כלשהו בשפה פרט ל " IF לוגי" ופרט למשפט "DØ" (ראה פרק 6).

המשמעות - אם הבטוי הלוגי 1 שבסוגריים הוא "נכון", יבוצע המשפט S והמחשב ימשיך בפקודה הבאה.

אם 1 הוא "לא נכון", ימשיך המחשב בפקודה הבאה ולא יבצע את הפקודה S .

<u>דוגמא:</u>

```
I = 0
IF (A.GE.0) I = 1
  :
```

אם $A \geq 0$ יהיה $I = 1$ אחרת יהיה $I = 0$.

<u>דוגמאות:</u>

```
IF (A.LT.5.1) GØ TØ 3
IF (K +J .LT. K*2) STØP
```

<u>דוגמא:</u> חישוב הפונקציה

$$Y(X) = \begin{cases} X + 5 & X \leq 3.5 \\ X - 5 & X > 3.5 \end{cases}$$

```
read X
IF (X .LE. 3.5) Y = X + 5
IF (X .GT. 3.5) Y = X - 5
print Y
STØP
END
```

<u>תרגיל:</u> חישוב הפונקציה

$$Y(X) = \begin{cases} -1 & 0 \leq X < 20 \\ 0 & 20 \leq X < 40 \\ +1 & 40 \leq X < 50 \end{cases}$$

<u>רמז:</u> יש לקרוא את X ולבדוק אם הוא בתחום בין אפס לחמשים. אם לא, יש לעצור. אחרת יש לבדוק אם $X \geq 20$ אם לא, יש לחשב $Y = -1$ ולהדפיס את Y . אחרת, יש לבדוק האם $X \geq 40$, אם לא, יש לחשב $Y = 0$ ולהדפיס את Y . אם כן, יש לחשב $Y = 1$ ולהדפיס את Y .

פרק 5

משתנים בעלי אינדקסים

עד עכשיו טפלנו במשתנים שכל אחד מהם הכיל מספר יחיד. לעתים רוצים שמשתנה יחיד יכיל הרבה מספרים וקיימת אפשרות כזו בשפה. אפשר להגדיר כל משתנה כמשתנה בעל אינדקס, ואז שמו של המשתנה מיצג כמה ערכים מספריים.

ההגדרה נעשית ע"י פקודה מיוחדת הנקראת "DIMENSIØN".

צורתה הכללי DIMENSIØN m_1 (ממד), m_2 (ממד)...

כאשר ...m_1 m_2 הם שמות של משתנים, והממדים הם קבועים שלמים חיוביים.

לדוגמא: .DIMENSIØN X (13), Q (45), L (111)

פקודה זו מגדירה את X Q L כמשתנים בעלי אינדקסים. X מכיל בתוכו 13 מספרים, Q יכול להכיל 45, L יכול להכיל 111 מספרים.

הפקודה DIMENSIØN מגדירה משתנים בעלי אינדקסים והגדרה זו אינה ניתנת לביטול. משתנה שהופיע בפקודה DIMENSIØN , כלומר שהוגדר כמשתנה בעל אינדקס, יהיה לאורך כל התכנית משתנה בעל אינדקס ואי אפשר יהיה להשתמש בו כבמשתנה פשוט.

הפקודה DIMENSIØN שונה עקרונית מכל הפקודות שראינו עד כה. זו פקודה שהמחשב אינו מבצע אותה. כאשר המחשב מגיע לפקודה זו הוא "רושם לפניו" שמשתנים מסוימים הם בעלי אינדקסים והוא מתיחס אליהם כך לאורך כל התכנית. הוא אינו מבצע כל פעולה לאחר קריאת פקודה זו ולכן נאמר שזו פקודה שאינה לביצוע.

פקודת ה- DIMENSIØN חייבת להופיע בראש התכנית לפני כל הפקודות הרגילות (פקודות לבצוע).

בהקשר לפקודה DIMENSIØN ראוי להזכיר את הפקודות INTEGER REAL . גם הן פקודות שאינן לביצוע ותפקידן הוא להקנות מעמד מסוים (מעמד של שלם או של ממשי) למשתנים מסוימים.

לדוגמא: הפקודה INTEGER X, A, K הופכת את המשתנים X A K למשתנים שלמים והם ישארו שלמים לכל אורך התכנית. (K הוא מראש משתנה שלם ולכן מיותר להכניס אותו לפקודה INTEGER, אולם זו אינה שגיאה).

הפקודה REAL L(50), J3 הופכת את המשתנים L J3 למשתנים ממשיים וכן הופכת את L למשתנה בעל אינדקס. במקרה כזה אין להכניס את L גם לפקודה DIMENSIØN.

הפקודות INTEGER REAL הן פקודות שאינן לביצוע וחיבות להופיע בתחילת התכנית.

כאשר משתנה X מוגדר כמשתנה בעל אינדקס יש להתיחס אליו בתכנית כאל משתנה בעל אינדקס. יש לציין את שמו וכן את האינדקס שלו. האינדקס אומר לאיזה מבין המספרים המהוים את המשתנה, מתכונים.

לדוגמא: DIMENSIØN X (50)

כאשר רוצים להקנות ערך מספרי למשתנה X יש לציין לאיזה מבין 50 ערכי X מתכונים. הפקודה X (3) = 1. תכניס את הערך 1. למקום השלישי במשתנה X . הפקודה

X = 1 אינה חוקית כי לא מצוין בה לאיזה מבין ערכי X מתכונים.

בפקודה DIMENSIØN חייב האינדקס להיות מספר שלם אולם כאשר משתמשים במשתנה לאורך התכנית, אפשר להשתמש באינדקס שהוא בעל מבנה מסובך יותר.

פקודה כמו X (I) = 15.3 היא חוקית בתנאי שהמשתנה I מוגדר ומכיל מספר בין 1 ל 50 (מספר שהוא בתחום של המשתנה X). הפקודה X (5*I - 13) = Y גם היא חוקית בתנאי שהבטוי האריתמטי 5*I - 13 מקבל ערך בין 1 ל 50.

באופן כללי יכול האינדקס להיות מורכב מקבועים שלמים, משתנים שלמים פשוטים (חסרי אינדקסים) או צרוף של אלה.

המשפט X (Z) = 5. לא חוקי מכיון ש- Z ממשי ולכן לא יכול לשמש כאינדקס. Integer

המשפט X (I (3)) = 5. אינו חוקי מכיון ש- I הוא משתנה בעל אינדקס ולכן אינו יכול לשמש בעצמו כאינדקס.

דוגמא: נתון משתנה X בעל 20 מקומות ויש לסכם את 20 המספרים הנמצאים במשתנה זה. התכנית תהיה:

```
  DIMENSIØN  X (20)
  read  X
  I = 1
  SUM = 0
5 SUM = SUM + X (I)
  I = I + 1
```

```
IF (I. LE. 20) GØ TØ 5
print SUM
STØP
END
```

הסבר:

כאשר המחשב מגיע בפעם הראשונה לפקודה SUM = SUM + X (I) נמצא בתוך I המספר 1 ובתוך SUM יש אפס. לכן ערכו של אגף ימין הוא 0 + X (1) וערך זה נכנס למשתנה SUM .

בפעם השניה נמצא בתוך I המספר 2 ובתוך SUM יש X (1) ולכן ערכו של אגף ימין הוא X (1) + X (2) וערך זה נכנס למשתנה SUM .

בפעם העשרים נמצא בתוך I המספר 20 ובתוך SUM יש X(1) + X (2) + . . . X (19) . לכן ערכו של אגף ימין הוא X(1) +. . .+ X (19) + X (20) וערך זה נכנס למשתנה SUM .

לאחר הפעם העשרים לא מתקיים התנאי I. LE. 20 ולכן לא מתבצעת הפקודה GØ TØ 5 . בפעם זו יעבור המחשב לפקודה print , ידפיס את התוצאה ויעצור.

<u>הערות:</u>

1) בפקודות read, print הופיע המשתנה X ללא אינדקס. דבר זה יוסבר בפרק 7.

2) הרעיון בתכנית הוא לחזור על אותה פקודה (הפקודה SUM = SUM + X (I)) 20 פעם. זה נעשה ע"י הגדלת

האינדקס I באחד ובדיקה אם הוא קטן או שוה 20.
אם כן, אזי חוזרים על החישוב, אם לא, העבודה נגמרת.

3) כאשר כותבים X(I) חיבים לדאוג לכך ש- I יכיל מספר בין 1 ל 20. כלומר I ימצא בתחום של X. דבר זה נעשה כאן מכיון ש- I מתחיל מאחד, גדל בקפיצות של 1 ואינו עולה על 20. בכל מקרה בו משתמשים במשתנה בעל אינדקס, יש לדאוג לכך שהאינדקס ימצא בתחום של המשתנה.

4) אם רוצים לסכם משתנה X המכיל לא 20 אלא 20000 מספרים יש לשנות רק שתי פקודות בתכנית.

```
DIMENSIØN X (20000), IF(I.LE.20000)GØ TØ 5
```

דבר זה מדגים את היתרון הגדול שיש בשימוש במשתנים בעלי אינדקסים.

מותר להשתמש גם במשתנים בעלי שנים או שלושה אינדקסים. לדוגמא הפקודה DIMENSIØN Y (35,4) , M (13,2,7) מגדירה משתנה Y בעל 140 אלמנטים שאפשר להסתכל עליו כעל מטריצה בעלת 35 שורות ו 4 עמודים. המשתנה M מכיל 182 מספרים ואפשר להסתכל עליו כעל אוסף של 7 מטריצות מסדר 2 × 13.

פרק 6

חזרות בתכנית, הפקודה DØ

בפרק הקודם הופיעה דוגמא של תכנית הכוללת חזרות.

התכנית חזרה 20 פעם על הקטע

```
5 SUM = SUM + X (I)
  I = I + 1
  IF (I. LE. 20) GØ TØ 5
```

מושג זה של חזרה על קטע בתכנית הוא בסיסי ושימושי מאוד, ולכן קימת פקודה מיוחדת בשפה, פקודה שבעזרתה אפשר לבצע בקלות חזרות על קטע בתכנית.

הפקודה נקראת DØ . היא בעלת שתי צורות כלליות

DØ n i = m_1, m_2, m_3

DØ n i = m_1, m_2

כאשר n - מספר זהוי של משפט בתכנית.

i - שם של משתנה שלם כלשהו.

m_1 m_2 m_3 - קבועים שלמים חיוביים או משתנים שלמים פשוטים (חסרי אינדקסים). Integer

<u>לדוגמא:</u>

```
DØ 5 J = 1, 10, 2
DØ 317 KK = 31, M, L
DØ 60 L1 = 1, 5
```

אם m_3 אינו מופיע אז המחשב מניח שערכו הוא 1.

המשמעות: המחשב יבצע את שורת הפקודות, החל מהפקודה שאחרי ה- DØ ועד הפקודה שמספרה n (כולל אותה),

מספר פעמים, כאשר המשתנה i מקבל את הערכים מ- m_1
עד m_2 בקפיצות של m_3.
המשתנה i יקבל את הערכים

$$m_1, m_1 + m_3, m_1 + 2\, m_3 \ldots, m_2$$

שורת הפקודות החל מהפקודה שאחרי ה- DØ ועד הפקודה שמספרה n, נקראת הטוח של ה- DØ.

דוגמא: שימוש בפקודה DØ לסיכום שורת מספרים.

```
  DIMENSIØN X(10)
  read X
  SUM = 0
  DØ 2 I = 1, 10
2 SUM = SUM + X(I)
  print X
  STØP
  END
```

דוגמא זו מראה כיצד השימוש בפקודה DØ מפשט את התכנית.

הטוח של ה- DØ כולל כאן פקודה אחת בלבד.
2 SUM = SUM + X(I) והיא מתבצעת עשר פעמים כאשר בכל פעם גדל I ביחידה אחת.

ע"י שנוי של פקודה אחת בתכנית שלמעלה, אפשר לקבל תכנית שתסכם לא את כל שורת המספרים, אלא רק את המספרים שבמקומות האיזוגיים, (המספרים הראשון, השלישי, החמישי, השביעי והתשיעי).
השנוי הוא DØ 2 I =1 ,10 ,2

כעת יעלה I מאחד עד עשר בקפיצות של שנים. בקפיצות אלה יקבל I את הערכים 1,3,5,7,9. I לא יכול להגיע

לערך 10 ולכן יעצר בערך 9 שהוא הקרוב ביותר ל-10 (ראה כלל 6 להלן).

כללים לשימוש ב- DØ.

1) הפקודה הראשונה בטוח של DØ חייבת להיות פקודה לבצוע, כלומר אסור שתהיה INTEGER DIMENSIØN או משהו דומה.

2) מותר להכניס טוח של DØ בתוך טוח של DØ אחר, אולם רק בצורה  או בצורה .

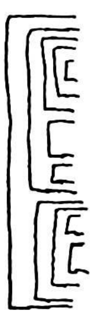
<u>לדוגמא:</u>

```
      DØ  100  K = 1, 5
      DØ  3     I = J, 10
    3 - - -
  100 - - -
```

```
      DØ   30  I = 1, 10
      DØ  30  J = 1, 15
   30 - - - -
```

אסור להכניס טוח אחד בתוך טוח שני בצורה .

למעשה מותר ליצור רשת שלמה של טוחים אחד בתוך השני כמו בדוגמא:



3) אסור שהפקודה האחרונה בטוח של DØ תהיה אחת מהבאות

GØ TØ פשוט או מחושב

IF אריתמטי או לוגי

DØ

STØP

FØRMAT

END

RETURN

CALL

אם בכל זאת צריכה פקודה כזו להיות אחרונה בטוח של DØ (פרט לפקודה END המופיעה רק בסוף התכנית) אפשר לרשום אותה במקומה ולסים את הטוח בפקודה CØNTINUE.

פקודה זו אינה מבצעת שום דבר. היא אומרת למחשב להמשיך בביצוע התכנית והיא נוצרה רק כדי לאפשר הכנסת פקודות מסוג GØ TØ, IF, CALL בסוף טוח של DØ.

דוגמא: הטוח הבא אינו חוקי

```
      DØ  100  N = 3, 15
100 IF (X(N).LT.0)G0 TØ 50
```

ולא יתבצע ע"י המחשב.

התיקון חייב להיות

```
      DØ 100 N = 3, 15
      IF (X(N).LT.0)GØ TØ 50
100  CØNTINUE
```

כעת פקודת ה- IF אינה אחרונה בטוח ולכן הטוח חוקי.

4) בתוך הטוח של DØ מותר להשתמש במשתנה i (וכן ב- m_1 m_2 m_3 אם הם משתנים) אולם אסור לשנותו.

לדוגמא:

```
DØ 5 I = 1, M
IF (I.EQ.3) A = B        שימוש במשתנה I , מותר
  .
  .
X = M                    שימוש במשתנה M , מותר
  .
  .
I = 5                    שנוי של I , אסור
  .
  .
read M                   שנוי של M , אסור
  .
  .
5 CONTINUE
```

5) על מנת לחזור על הטוח של ה- DØ מספר פעמים כמבוקש, יש להיכנס לתוך הטוח דרך הפקודה DØ .
כניסה כבדוגמא הבאה היא לא חוקית.

```
   .
   .
   GØ TØ 3
   DØ 50 I = 1, 2
   .
   .
3  - . -
   .
   .
50 - - -
```

לעומת זאת, כאשר נמצאים בתוך טוח של DØ , אפשר לקפוץ החוצה ולחזור בחזרה. לאחר קפיצה כזו ממשיך הטוח להתבצע.

דוגמא:

```
      DØ  5  I = 1, 10
      .
      .
      GØ  TØ  30
      .
      .
10    - --
      .
      .
 5    CØNTINUE
      .
      .
30    - - -
      .
      .
      GØ TØ 10
```

6) המשתנה i עולה מ- m_1 בקפיצות של m_3 ולא תמיד יכול להגיע ל- m_2 . במקרה כזה יגיע i לערך הקרוב ביותר ל- m_2 (וקטן מ- m_2) אליו הוא יכול להגיע.

לדוגמא:

```
DØ  1  K = 1, 14, 5
```

K יקבל את הערכים 11, 6, 1.

דוגמאות לשימוש בפקודה DØ .

1) נתונה שורה של 20 מספרים בתוך משתנה V . יש למצוא את המספר הקטן ביותר מבין ה-20 ולהחליפו עם המספר הראשון מביניהם. במלים אחרות - יש להעביר את המספר המינימלי למקום הראשון ותוך כדי כך אולי להחליף את המקומות של מספרים אחרים.

התכנית תהיה

```
DIMENSIØN  V (20)
read  V
DØ  30  J = 2,  20
IF (V (1). LE. V (J)) GØ TØ 30
T = V (1)
```

```
      V (1) = V (J)
      V (J) = T
   30 CØNTINUE
      STØP
      END
```

הסבר: המספרים $V(2), \ldots, V(20)$ מושווים עם המספר הראשון $V(1)$ ע"י "IF לוגי". אם המספר הראשון $V(1)$ קטן (או שוה) מהמספר הנוכחי $V(J)$ אזי הולכים לפקודה 30 (כלומר לסוף הטוח של ה- DØ) וממנה חוזרים אוטומטית לתחילת ה- DØ.

אם המספר הראשון $V(1)$ גדול מהמספר הנוכחי $V(J)$ אזי שני המספרים מוחלפים, והקטן מביניהם נעשה הראשון. ההחלפה נעשית ע"י שלושת הפקודות

```
T = V (1)
V (1) = V (J)
V (J) = T
```

המשתמשות במשתנה T הממלא כאן תפקיד של משתנה ביניים.

בכל מקרה, כאשר מגיעים לפקודה 30 בטוחים שבמקום הראשון נמצא המספר הקטן מבין שני המספרים שהושוו. לאחר תום כל ההשואות, בטוחים שבמקום הראשון נמצא המספר הקטן ביותר מכולם.

2) נתונה שורה של 20 מספרים בתוך משתנה V. יש למיין אותם בסדר עולה.

התכנית היא

```
      DØ 30 I = 1, 19
      I1 = I + 1
```

```
      DØ  30 J = I1, 20
      IF (V(I).LE.V(J)) GØ TØ 30
      T = V(I)
      V (I) = V (J)
      V (J) = T
30    CØNTINUE
      STØP
      END
```

הסבר: התכנית מכילה שני טוחים של DØ , אחד בתוך השני. הפנימי מביניהם זהה בצורתו ל- DØ שבדוגמא הקודמת אלא שפה כתוב I במקום 1. בשלב הראשון של ה- DØ החצוני יקבל I את הערך 1 ולכן ה- DØ הפנימי יהיה זהה לזה שבדוגמא הקודמת.

פעולת ה- DØ הפנימי תהיה לכן העברת המספר המינימלי למקום הראשון.

בשלב השני של ה- DØ החצוני יקבל I את הערך 2 ולכן ה- DØ הפנימי יקבל את הצורה הבאה.

```
      DØ  30  J = 3, 20
      IF (V(2).LE.V(J)) GØ TØ 30
      T = V (2)
      V (2) = V (J)
      V (J) = T
30    CØNTINUE
```

כלומר זה יהיה קטע של תכנית שיבדוק את המספרים במקומות 2,3,4,...,20 ויעביר את המינימלי מביניהם למקום השני.

בשלב השלישי של ה- DØ יעבור למקום השלישי המספר שהוא המינימלי מבין 18 האחרונים.

בצורה כזו משיגים מיון של 20 המספרים כך שהמינימלי הוא במקום הראשון והמקסימלי במקום העשרים.

3) יש לחשב את הבטוי $\sqrt{9 - X^2}$ כאשר X מקבל את הערכים 3...2.2 2.1 2 ולהכניס את 11 התוצאות לתוך משתנה Y בעל 11 מקומות.

התכנית היא

```
      DIMENSIØN Y (11)
      DØ  10  I = 20, 30
      X = I
      X = X/ 10.
10    Y (I - 19) = (9 - X **2) ** .5
```

הסבר: I מקבל את הערכים 30...20,21 הפקודה X = I גורמת לכך ש- X יקבל את הערכים .30... .21 ,.20 הפקודה X = X/10. גורמת לכך שבסופו של דבר יהיו ב-X הערכים .3 ... 2.2 2.1 .2 שהם הערכים הרצויים.

את תוצאות החישוב מכניסים לתוך (I - 19) Y ולא לתוך (Y(I וזה מכיון ש- I עולה מ-20 עד 30 ואילו Y מכיל את האלמנטים (11) Y -- (2) Y ,(1) Y . האינדקס 19- I משתנה מ-1 עד 11 ומביא לכך לתוצאה הרצויה.

4) כאשר רוצים לכתוב DØ שבו המשתנה קטן בקפיצות (במקום לגדול) אפשר להשיג זאת ע"י הוספת פקודה יחידה, לדוגמא

```
DØ 7 I = 1, 10
I1 = 11 - I
  .
  .
  .
```

I1 הוא משתנה שירד מ-10 עד 1 בקפיצות של 1.

<u>תרגילים:</u>

1) נתונים שני משתנים W (100) , Y (100) לכתוב תכנית שתחשב את $\sum$ Y (I) W (I) גודל זה נקרא המכפלה הסקלרית של הוקטורים W Y .

2) נתונים המשתנים B (3,7) , A (3,7) לכתוב תכנית שתכניס לתוך A(I, J) את הסכום B (I, J) + A(I, J) פעולה זו נקראת חבור המטריצות B A, .

3) נתונים המשתנים C (5,8) , B (6,8) , A (5,6) לכתוב תכנית שתכניס לתוך C (I, J) את $\sum_{K=1}^{6}$ A(I, K) * B(K, J) פעולה זו נקראת כפל המטריצות B A, .

פרק 7

קלט - פלט

כל הפקודות שנלמדו עד עכשיו הן בעלות מכנה משותף אחד. אלה הן פקודות פנימיות, פקודות המבוצעות בתוך המחשב. ביצוען או אי ביצוען של פקודות אלה אינו משנה דבר מסביב למחשב.

יש לכן צורך בפקודות נוספות, פקודות שתהיינה חיצוניות, כלומר שתקשרנה בין המחשב למה שנעשה סביבו. הקשר בין המחשב לעולם החצוני מתבטא בקריאת נתונים מבחוץ לתוך המחשב ובהדפסת תוצאות מהמחשב החוצה.

קשר זה מתבצע באמצעות שתי הפקודות READ PRINT.

הפקודה READ קוראת נתונים מבחוץ (צורת הגשת הנתונים לתכנית תתואר בפרק 9) ומכניסה אותם לתוך משתנים של התכנית. הפקודה PRINT מדפיסה את ערכיהם של משתנים מהתכנית ויכולה גם להדפיס כותרות.

הצורה הכללית של הפקודות היא

READ n, list

PRINT n, list

n היא מספר זהוי של פקודה הנקראת פקודת FØRMAT ושתתואר להלן.

list היא רשימה של משתנים המופרדים ע"י פסיקים, רשימה היכולה לכלול DØ פנימי.

דוגמאות: READ 5, A, I, X

זו פקודה הקוראת שלושה מספרים מכרטיס נתונים (שהתכניתן

חייב לספק אותו יחד עם התכנית), מכניסה את הראשון למשתנה A , את השני למשתנה I ואת השלישי למשתנה X .

לאחר בצוע הפקודה יהיו במשתנים A I X המספרים החדשים, וכל מה שהיה בהם קודם, יעלם.

בדוגמא זו ה- list הוא A, I, X וחייבת להיות פקודת FØRMAT שמספרה 5.

הפקודה PRINT 13, M (3), J, Z מדפיסה (לפי FØRMAT מספר 13) את המשתנה M(3) (האלמנט השלישי של המשתנה M), את המשתנה J ואת המשתנה Z .

חשוב להבין שקריאת נתונים מבחוץ נעשית אך ורק לתוך משתנים. הדפסת תוצאות החוצה יכולה להיעשות או מתוך משתנים או בצורה של הדפסת כותרות (דבר זה יוסבר בפירוט בסעיף על פורמט H).

לתוך הפקודות PRINT READ אפשר להכניס DØ פנימי כפי שמראות הדוגמאות הבאות

READ 77, (A (I), I = 1, 3)

כאן יקראו שלושה מספרים שיוכנסו לשלושת המקומות הראשונים. במשתנה A .

PRINT 8, Z, (K (J), J = 2, 4), K(15)

כאן יודפסו (לפי הסדר) Z K(2) K(3) K(4) K(15).

ה- DØ הפנימי סגור בתוך סוגריים, מכיל רשימה של משתנים, אח"כ פסיק ואח"כ בטוי מהצורה $L = m_1, m_2, m_3$ כמו ב- DØ הרגיל.

דוגמא: READ 5, (A (I), B (I), C (I), I = 1, 10, 2)

הנתונים יקראו לתוך המשתנים הבאים (משמאל לימין)

A (1) B(1) C(1) A(3) B(3) C(3) - - -
- - - A(9) B(9) C(9)

דוגמא: PRINT 1, (A(I), I = 1, 2), (A(I), I = 5, 6)

כאן קיימים שני DØ פנימיים המופרדים אחד מהשני. קודם מתבצע כל ה DØ הראשון ואח"כ כל השני. המספרים שיודפסו יהיו (משמאל לימין) A(1) A(2) A(5) A(6)

יתכנו גם כמה DØ פנימיים הנמצאים אחד בתוך השני.

ה- DØ הראשון מכיל רשימה של משתנים, אח"כ פסיק ואח"כ $L = m_1, m_2, m_3$.

במקום אחד מהמשתנים שברשימה אפשר לרשום שוב DØ פנימי שני.

דוגמא: READ 2, (A (I), (B(I, J), J =1, 2), C(I), I = 1, 3)

המשתנה השני שברשימה מוכל בתוך DØ פנימי והמספרים שיקראו מבחוץ יוכנסו למשתנים הבאים (משמאל לימין)

A(1) B(1, 1) B(1, 2) C(1)

A(2) B(2, 1) B(2, 2) C(2)

A(3) B(3, 1) B(3, 2) C(3)

דוגמא: PRINT 81, ((X (K, L), K = 1, 2), L = 1, 3)

כאן קיימים שני DØ פנימיים, אחד בתוך השני והמספרים שיודפסו הם

X(1, 1) X(2, 1) X(1, 2) X(2, 2) X(1, 3) X(2, 3)

בפקודה READ אפשר לקרוא מספר לתוך משתנה ולהשתמש במשתנה זה כבר באותה הפקודה.

דוגמא: READ 2, M, (A(J), J = 1, M)

באותה פקודה קוראים קודם מספר לתוך המשתנה M ואח"כ משתמשים באותו משתנה בתור גבול עליון של DØ פנימי.

פורמט

כאשר קוראים מספרים מכרטיס, יש לפרט למחשב באילו עמודות בכרטיס מופיע כל מספר, מה מספר הספרות שלו ואם הוא אינו שלם, כמה ספרות אחרי הנקודה העשרונית הוא מכיל.

כאשר מדפיסים מספרים יש לפרט למחשב באיזה מקום על הניר להדפיס כל מספר וכמה ספרות להקדיש לו.

כל הפרוטים האלה נתנים למחשב ע"י פקודת הפורמט. פקודה זו אינה גורמת לביצועה של פעולה כלשהי, אלא היא מהווה אוסף של פרוטים, של ספציפיקציות למחשב, איפה ואיך להדפיס או מהיכן וכמה לקרוא. לכן נחשבת גם פקודה זו בין הפקודות שאינן לביצוע.

דוגמא: שלושה מספרים מנוקבים על כרטיס נתונים בצורה

- 123.456137.89102.57

כאשר מגישים כרטיס זה למחשב, הוא אינו מסוגל לנחש איפה מתחיל ונגמר כל מספר ולכן יש לפרט לו זאת. אפשר למשל לומר לו שהמספר הראשון מתחיל מתחילת הכרטיס, ארכו שמונה מקומות והוא מכיל שלוש ספרות אחרי הנקודה העשרונית (הנקודה וסימן המינוס נכללים בספירת המקומות). המספר השני מתחיל מיד אחרי הראשון, ארכו ששה מקומות והוא בעל שתי ספרות אחרי הנקודה העשרונית. השלישי מתחיל מיד לאחר השני ומבנהו זהה לזה של השני.

אינפורמציה זו נכללת בפקודת הפורמט ובעזרתה קורא המחשב את שלושת המספרים 102.57 137.89 123.456 - מהכרטיס ומכניס אותם למשתנים המתאימים.

הצורה הכללית של הפקודה היא

n FØRMAT (שדה, ... , שדה, שדה)

n – הוא מספר זהוי (חיב להופיע). הפורמט מורכב משדות המופרדים ע"י פסיקים וסגורים בסוגריים.

כל שדה מתיחס למספר אותו רוצים לקרוא או להדפיס והוא מפרט:

א) האם המספר שלם או ממשי.

ב) מה מספר המקומות שהוא תופס על הכרטיס או שיש להקציב לו על ניר ההדפסה.

ג) כמה מבין מקומות אלה הם אחרי הנקודה העשרונית.

<u>שדה I</u>

צורתו הכללית – Iw כאשר w מספר שלם חיובי.

משמעותו – המספר הבא שיקרא (או יודפס), הוא שלם ותופס w מקומות כולל הסימן. לדוגמא – את המספר 168- יש לקרוא (או להדפיס) בפורמט I4 .

דוגמא: כרטיס נתונים ועליו שלושה מספרים מנוקבים בצורה

1234567

שתי הפקודות

```
      READ 5, I, J, K
5     FØRMAT (I4, I1, I2)
```

תגרום לכך שהכרטיס יקרא לתוך המחשב, למשתנה I יכנס המספר 1234, למשתנה J יכנס 5 ולמשתנה K – 67.

פקודות אלה מבוצעות ע"י המחשב בשלבים הבאים:

1) הפקודה READ גורמת לקריאת הכרטיס.

2) המספר 5 גורם לכך שהמחשב יתיחס אל פקודה מספר 5 כאל פורמט ויתחיל לעבור על הפורמט משמאל לימין.

3) השדה הראשון הוא I4 ולכן יקח המחשב את התוכן של ארבע העמודות הראשונות על הכרטיס ויכניס אותו למשתנה הראשון I ברשימה.

4) השדה השני הוא I1 ולכן יקח המחשב את התוכן של העמודה הבאה (החמישית) בכרטיס ויכניס אותו למשתנה השני J ברשימה.

5) השדה השלישי הוא I2 ולכן יקח המחשב את התוכן של שתי העמודות הבאות (ששית ושביעית) בכרטיס ויכניס אותו למשתנה השלישי K ברשימה.

6) בשלב זה מגיע המחשב לסוגר הימני של הפורמט (המציין סוף פורמט) וכן לסוף רשימת המשתנים בפקודה READ . דבר זה מסמן למחשב שנגמר ביצועה של פקודת הקריאה, והוא עובר לפקודה הבאה בתכנית.

<u>שדה F</u>

צורתו הכללית - F w . d כאשר w d שלמים חיוביים המקימים $w > d \geq 0$.

משמעותו - המספר הבא שיקרא (או יודפס) הוא ממשי, ארכו w מקומות והוא מכיל d ספרות אחרי הנקודה העשרונית.

w המקומות יכולים לכלול סימן ונקודה עשרונית.

לדוגמא: כרטיס נתונים שעליו מנוקב

14.547 - 31.54444

שתי הפקודות

```
READ 17, X; Y, Z
17 FORMAT (F 6.3, F 5.1, F 4.0)
```

יגרמו לכך שלתוך המשתנה X יכנס המספר הממשי 14.547, לתוך Y יכנס 31.5 - ולתוך Z יכנס .4444.

הערה: כאשר d = o (כמו בשדה F 3.0) לא חיבת להופיע הנקודה העשרונית. המספר יכול להיות מנוקב על הכרטיס כמספר שלם (ללא הנקודה), אולם קריאתו בפורמט F גורמת לכך שהוא יכנס לתוך המשתנה כמספר ממשי (שאין לו ספרות אחרי הנקודה).

שלבי העבודה של המחשב במקרה זו יהיו זהים לשלבים שהוארו מקודם.

<u>שדה X</u>

צורתו הכללית wX , כאשר w שלם חיובי.

משמעותו – דילוג על w מקומות.

כאשר מופיע שדה זה בפורמט של קריאה, ידלג המחשב על המקומות המתאימים בכרטיס ולא יקרא אותם.

כאשר הוא מופיע בפורמט של הדפסה, ישאיר המחשב w מקומות ריקים בשורת ההדפסה וימשיך להדפיס משם.

דוגמא: כרטיס נתונים שעליו מנוקב

13.47b - 405.ABC 15 - 1

שתי הפקודות

```
   READ 20, G, H, I, N
20 FØRMAT (F 5.2, 1X, F 5.0, 3X, I2, 1X, I1)
```

תגרומנה לכך שהמספרים

	13.47	- 405.	15	1
יכנסו למשתנים	G	H	I	N

בהתאמה

שלבי העבודה של המחשב יהיו במקרה זה:

1) המלה READ גורמת לקריאת הכרטיס.

2) המספר 20 גורם לכך שהמחשב יתיחס אל פקודה מספר 20 כאל פורמט.

3) השדה הראשון F 5.2 גורם לכך שהמספר בעמודות 1-5 בכרטיס יכנס למשתנה הראשון G.

4) השדה השני 1 X גורם לדילוג על העמודה הבאה (עמודה 6) בכרטיס.

5) השדה השלישי F 5.0 גורם לכך שבתוכן של חמש העמודות הבאות (עמודות 7-11) יכנס למשתנה השני H.

6) השדה הרביעי 3 X גורם לדילוג על עמודות 12-14 (המחשב אינו קורא את האותיות המנוקבות שם).

7) השדה החמישי I 2 גורם להכנסת התוכן של עמודות 15-16 למשתנה I.

8) השדה השישי 1 X גורם לדילוג על עמודה 17 (בה מנוקב סימן מינוס).

9) השדה השביעי I 1 גורם להכנסת התוכן של עמודה 18 למשתנה N.

10) כעת מגיע המחשב לסוגר הימני של הפורמט וגם לסוף רשימת המשתנים. דבר זה גורם לסיום פעולת הקריאה.

דוגמא: המשתנים מהדוגמא הקודמת מודפסים עם רוחים ביניהם. הפקודות תהיינה

```
      PRINT 30, G, H, I, N
30    FØRMAT (1 X, F 5.2, 1 X, F 4.0, 1 X, I 2, 1X, I1)
```

כל אחד מהמספרים מודפס כשמצד שמאל שלו יש רוח אחד.
שורת ההדפסה נראית

b13.47b - 405b15b1

כאשר b פרושו מקום ריק על הניר.

הערה: המשתנה H נקרא בפורמט F 5.0 אולם מאחר ואינו מכיל ספרות אחרי הנקודה, הרי שמספיקים ארבעה מקומות על הניר בכדי להדפיסו. לכן אפשר להדפיסו בפורמט F 4.0 .

<u>שדה E</u>

צורתו הכללית - E w.d כאשר w d שלמים חיוביים המקימים $w > d \geqslant 0$.

משמעותו - המספר הבא שיקרא (או יודפס) הוא מספר ממשי, ארכו w מקומות, הוא מכיל d ספרות אחרי הנקודה והוא רשום בעזרת מעריך.

לדוגמא: יש לקרוא את המספר

$$-0.00\underbrace{\ldots\ldots}_{58}05713$$

אפשר לנקב אותו על כרטיס ולקרוא אותו בפורמט F 65.62 , אולם מאחר ורוב הספרות בו הן אפסים, הרי יותר נוח לנקב אותו על הכרטיס בצורה 58 - E 0.5713 - ולקרוא אותו בפורמט E 11.4 .

הכתיב 58 - E 0.5713 - פרושו $-0.5713\cdot10^{-58}$.

פורמט E מצפה למצוא על הכרטיס מספר ממשי שאחריו האות E ואחריה מעריך עשרוני.

במקרה של פורמט E מסמן w את האורך הכללי של המספר, כולל האות E והמעריך. d מסמן את מספר הספרות שאחרי הנקודה ואינו כולל את המעריך או את האות E .

פורמט E נוח לשימוש במספרים גדולים מאוד או קטנים מאוד, שרוב ספרותיהם הן אפסים.

<u>תכונות של שדות מספריים.</u>

שלושת השדות E F I הם מספריים, כלומר משמשים לקריאת והדפסת מספרים והם בעלי התכונות הבאות.

1) בקריאה של מספר מכרטיס, יכול הכרטיס להכיל ספרות, סימן וכן נקודה (במקרה של מספר ממשי) או האות E . במקרה שהכרטיס מכיל משהו אחר, יודיע המחשב על שגיאה ויפסיק את בצוע התכנית.

לדוגמא: כרטיס ובו מנוקב. 13 A 4
נסיון לקרוא כרטיס זה
בפורמט I4 , יגרום לכך שהמחשב יודיע על שגיאה ויעצור.

2) מקומות ריקים על הכרטיס אקויולנטיים לאפסים.

לדוגמא: הניקוב 1 b1
יקרא בפורמט I 3 בתור המספר 101.

3) בהדפסה, עלול השדה המוקצב למספר, להיות גדול מארכו של המספר. במקרה זה יודפס המספר בצד ימין של השדה וצד שמאל שלו ישאר ריק.

<u>לדוגמא:</u>

```
  Z = - 13.4
  PRINT 2, Z
2 FØRMAT (F 7.1)
```

המספר Z דורש להדפסתו רק חמשה מקומות ואילו הפורמט מקציב לו שבעה מקומות. לכן יודפס המספר כשמצד שמאל שלו שני מקומות ריקים. bb - 13.4 .

4) אם השדה המוקצב למספר, קטן מארכו של המספר, יודפס כוכב * בצד שמאל של השדה ויתר המקומות בשדה ימולאו בספרות המתאימות.

<u>לדוגמא:</u>

```
  Z = 704.5
  PRINT 3, Z
3 FØRMAT (F 4.1)
```

יודפס *4.5

5) בהדפסה של משתנה ממשי אין צורך להדפיס את כל הספרות שאחרי הנקודה. אפשר להדפיס רק חלק מהן ואז המחשב מדפיס את המספר כשהוא מעוגל.

לדוגמא:

```
  Z = 24.78654
  PRINT 4, Z
4 FØRMAT (F 5.2 )
```

יודפס המספר 24.79.

אם לעומת זאת מבקשים להדפיס יותר ספרות אחרי הנקודה מאשר יש למספר, יודפסו אפסים.

לדוגמא:

```
  Z = 24.7
  PRINT 4, Z
4 FØRMAT (F 5.2)
```

יודפס 24.70.

דוגמא: תכנית הקוראת שני כרטיסי נתונים, מבצעת חישובים על הנתונים שקראה ומדפיסה תוצאות.

```
  READ 3, A, B
  C = A + B
  D = A*B
  E = C/ D
  READ 5, I, J
  I1 = I + 1
  I2 = I * J
  PRINT 7, A, B, C, D, E
7 FØRMAT (F 6.2, F 6.2, F 7.2, F 10.2, F6.4)
```

```
      PRINT 8, I, J, I1, I2
    8 FØRMAT (I 4, I 4, I 6, I 8)
      STØP
    5 FØRMAT (I 3, I 3)
    3 FØRMAT (F 5.2, 4 X, F 5.2)
      END
```

<u>הערות:</u>

1) פקודת הפורמט אינה חיבת לבוא ליד פקודת READ/PRINT המתאימה, אלא יכולה להופיע בכל מקום שהוא בתכנית.

2) כאשר, במהלך ביצוע התכנית, מגיע המחשב בפעם הראשונה לפקודת READ כלשהי, הוא יקרא את כרטיס הנתונים הראשון. כאשר הוא מגיע בפעם השניה לפקודת READ כלשהי, הוא יקרא את כרטיס הנתונים השני וכו׳.

בדוגמא האחרונה הגיע המחשב קודם לפקודה READ 3, A, B ולכן קרא את כרטיס הנתונים הראשון לפי פורמט 3. כרטיס זה חייב לכן להכיל שני מספרים ממשיים, הראשון בעמודות 1-5 והשני בעמודות 10-14. בשאר העמודות יכול הכרטיס להכיל קרקטרים כלשהם.

פקודת ה- READ השניה שפוגש המחשב היא READ 5, I, J ולכן הוא יקרא את כרטיס הנתונים השני לפי פורמט מספר 5. כרטיס זה חייב לכן להכיל שני מספרים שלמים, הראשון בעמודות 1-3 והשני בעמודות 4-6. שאר העמודות יכולות להכיל קרקטרים כלשהם.

התכניתן שכתב את התכנית, יודע כמה פקודות READ היא

מכילה, לאיזו מהן מגיע המחשב ראשונה, לאיזו הוא מגיע שניה וכו׳. מכאן יודע התכניתן כמה כרטיסי נתונים תקרא התכנית ועל ידי איזו פקודת READ יקרא כל כרטיס. במלים אחרות, התכניתן יודע באיזה פורמט קוראת התכנית כל אחד מכרטיסי הנתונים, ועליו להכין את הכרטיסים כך שכל אחד מהם יתאים לפורמט לפיו הוא נקרא. על כרטיס הנתונים אין צורך לנקב את מספר פקודת הפורמט המתאימה, או את מספר פקודת ה- READ או כל דבר אחר. על כרטיס הנתונים יש לנקב רק את הנתונים, וזאת לפי הפורמט בו יקרא הכרטיס.

אם כרטיס נתונים נקרא לפי הפורמט (35X, I3) יש לנקב בו את המספר בעמודות 36-38.

בכל מקרה יש להתאים את כרטיס הנתונים לפורמט לפיו הוא נקרא. אין צורך להתחיל לנקב את הכרטיס מעמודה 1 או מעמודה 7 או מכל עמודה אחרת.

<u>שדה H</u>

צורתו הכללית $wHa_1 \ldots a_w$ כאשר w שלם חיובי, $a_1 \ldots a_w$ קרקטרים כלשהם (כולל בלנק).

דוגמא: 3HABC   .4H .lb $

המשמעות: w הקרקטרים המופיעים אחרי H יודפסו על הניר כמות שהם.

שדה זה משמש להדפסת כותרות. אפשר להדפיס בעזרתו כל קרקטר שהוא, אולם הוא אינו שדה מספרי כי אי אפשר להדפיס בעזרתו ערכים של משתנים.

דוגמא:

```
  S = 15.
  PRINT 1, S
1 FϕRMAT (3HbN = , F 2.0)
```

יודפס bN = 15.

יש לשים לב לכך שגם בלנק נספר בין הקרקטרים שאחרי H וזה יוצא מן הכלל לחקי הכתיבה של השפה האומרים שאין חשיבות לבלנקים.

<u>הזזת שורה על הניר</u> <u>(carriage control).</u>

כל פקודת PRINT מתחילה להדפיס מתחילת שורה חדשה. לעתים רוצים להשאיר רוח כפול בין השורה הנוכחית לקודמת, או רוצים להדפיס את השורה הנוכחית בתחילת דף חדש.

לא קיימות פקודות מיוחדות בשפה לשם כך, ומבצעים זאת בעקיפין. הדבר האומר למחשב איזה רוח להשאיר לפני הדפסת השורה הנוכחית, הוא הקרקטר הראשון של שורה זו.

הקרקטר הראשון של כל שורה אינו מודפס והמחשב מפרש אותו כהוראה האומרת מה המרחק של שורה זו מהשורה הקודמת. קיימות ארבע אפשרויות.

1) הקרקטר הראשון הוא +. במקרה זה לא יהיה רוח בין השורה הנוכחית לשורה הקודמת.

2) בקרקטר הראשון הוא בלנק. במקרה זה יהיה רוח רגיל בין השורה הנוכחית והקודמת.

3) הקרקטר הראשון הוא אפס. במקרה זה יהיה רוח כפול בין השורה הנוכחית והקודמת.

4) הקרקטר הראשון הוא 1. במקרה זה תודפס השורה הנוכחית בתחילת עמוד חדש ושארית העמוד הקודם תישאר ריקה.

אם הקרקטר הראשון בשורה שונה מארבעת אלה שצוינו, ידלג המחשב על מספר שורות לפני הדפסת השורה הנוכחית, ודילוג זה יבוצע לפי "שרירות לבו" של המחשב, כלומר אין לדעת על כמה שורות ידלג.

<u>דוגמא:</u>

```
    PRINT 4
4   FØRMAT ( 8 H1 KØTERET)
```

יודפס KØTERET בתחילת עמוד חדש. הספרה 1 לא תודפס מכיון שהמחשב מפרש אותה כהוראה לקפוץ לעמוד חדש.

<u>דוגמא:</u>

```
  Z = 103.1
  PRINT 1, Z
1 FØRMAT (F 5.1)
```

יודפס b 3.1 בתחילת עמוד חדש. זה מכיון שהמחשב מפרש את הספרה 1 כהוראה לקפוץ לעמוד חדש.

הפורמט הנכון הוא 1 FØRMAT (1 X , F 5.1)

השדה 1 X יוצר בלנק בתחילת השורה וזה אומר למחשב להשאיר רוח רגיל בין שורה זו לשורה הקודמת.

יש לכן להקפיד שכל פורמט להדפסה לא יתחיל בשדה נומרי אלא ברוח (שדה X) או בכותרת (שדה H).

<u>הערה:</u> מספר הקרקטרים המכסימלי שאפשר להדפיס בשורה, שונה ממחשב למחשב. במחשב שבאוניברסיטה מספר זה הוא 136. מכיון שהקרקטר השמאלי אינו מודפס הרי שנשארים לכל היותר 135 מקומות בשורה. אם מנסים להדפיס יותר מזה, יודיע המחשב על שגיאה ויפסיק את העבודה.

גורם חזרה (Repetition Factor)

אפשר בצורה נוחה לחזור על שדה (או קבוצת שדות) המופיע בפורמט.

במקום הפורמט (F 6.1, F 6.1, F 6.1)

אפשר לכתוב (3 F 6.1).

המספר 3 נקרא גורם חזרה.

במקום הפורמט (1X, F2.0, I3, 3X, F2.0, I3, 3X, I5)

אפשר לכתוב (1X, 2 (F2.0, I3, 3X), I5)

כאן גורם החזרה הוא 2 ולכן יחזור המחשב פעמיים על כל השדות הרשומים בסוגריים, ורק לאחר מכן ימשיך לשדה I 5.

יכולים להיות כמה גורמי חזרה בפורמט, אולם עליהם להופיע אחד אחרי השני ולא אחד בתוך השני.

מותר לכתוב FØRMAT (- -, 3(- -), - -4 (- -) - -)

ואסור לכתוב FØRMAT (- -, 2 (- - -, 5 (- -) - -) - -)

רקורד חדש

רקורד חדש מסומן בפורמט ע"י קו נטוי /.

המשמעות - בקריאה, לעבור לכרטיס הבא.

בהדפסה, לעבור לתחילת השורה הבאה.

דוגמא:

```
   PRINT 15, A, I
15 FØRMAT (3HbA = , 3X, 2HI = /1X, F4.1, I4)
```

ההדפסה תהיה בת שתי שורות

```
A=        I=
12.2 - 441
```

דוגמא:

שתי הפקודות READ 31, A, B, X, Y

31 FØRMAT (2 F 10.3 / 2 F 10.3)

יגרמו לכך שערכי המשתנים A, B יקראו מכרטיס אחד, וערכי המשתנים X, Y יקראו מהכרטיס הבא.

תכונות נוספות של פקודת הפורמט.

1) כאשר קוראים מספרים בפורמט E או F, לא חייבת להופיע נקודה על הכרטיס. במקרה כזה קובעים ערכי w d.

לדוגמא: כרטיס ועליו מנוקב 6901

אפשר לקרוא אותו בפורמט

F 4.1 ואז הוא יקרא בתור 690.1.

בפורמט F 4.2 הוא יקרא בתור 69.01.

בפורמט F 4.4 הוא יקרא בתור 0.6901.

כאשר קיימת אי התאמה בין ערכי w, d לבין מיקומה של הנקודה על הכרטיס, קובע מיקומה של הנקודה.

לדוגמא: כרטיס ובו מנוקב 69.01

אפשר לקרוא בפורמט F 5.2

ואז יקרא המספר 69.01. אפשר גם לקרוא אותו בפורמט F 5.3 או F 5.5 וכו'. בכל המקרים יקרא המספר 69.01.

2) אם קיים DIMENSIØN A (10)

אז הפקודה READ 3, (A(I), I = 1, 10)

תקרא 10 מספרים לתוך המשתנה A. במקרה כזה אפשר לכתוב READ 3, A ופקודה זו תעשה אותו דבר.

כלומר כאשר רוצים לקרוא לתוך כל האלמנטים של משתנה מסוים, אפשר לותר על ה- DØ הפנימי.

לגבי משתנה בעל שני ממדים קיים דבר דומה.

לדוגמא: אם DIMENSIØN X (10, 3)

אז שתי הפקודות

```
PRINT 10, ((X (I, J), I = 1, 10), J = 1, 3)
PRINT 10, X
```

הן זהות.

כלומר הפקודה PRINT 10, X מדפיסה את כל האלמנטים של המשתנה X כאילו היא כוללת שני DØ פנימיים שמהם הפנימי הוא על האינדקס הראשון, והחצוני, על האינדקס השני.

<u>הקשר בין ליסט ופורמט.</u>

כאשר קיימת אי התאמה בין ארכה של רשימת המשתנים (list) ובין ארכו של הפורמט, קובע ארכו של הליסט.

א: אם הליסט קצר מהפורמט, לא יגיע המחשב לסוף הפורמט.

דוגמא:

```
READ 16, A, B, C
16FØRMAT (2 F 3.1, F 4.2, F 2.0)
```

הליסט מכיל שלושה משתנים ואילו הפורמט מכיל ארבעה שדות. במקרה זה יהיו שלבי העבודה של המחשב כדלקמן:

1) המלה READ גורמת לקריאת כרטיס.
2) המספר 16 גורם לכך שהמחשב יתיחס אל פקודה מספר 16 כאל פורמט.
3) שני השדות הראשונים 2 F 3.1 יותאמו למשתנים A, B .
השדה השלישי F 4.2 יותאם למשתנה C .

4) לאחר קריאת המספר המתאים לתוך המשתנה C , מגיע המחשב לסוף רשימת המשתנים וזה מציין לו את גמר פעולת הקריאה.

המחשב לא יגיע לכן לשדה הרביעי F 2.0 ושדה זה לא ישפיע על פעולת הקריאה.

ב: אם הליסט ארוך מהפורמט, יבצע המחשב את פקודת הקלט/פלט בצורה רגילה עד שיגיע לסוף הפורמט. בנקודה זו יבצע המחשב שתי פעולות.

1) יעבור לרקורד חדש (כלומר יקרא כרטיס חדש או יעבור לשורה חדשה).

2) יחזור לתחילת הפורמט.

לאחר בצוע פעולות אלה ימשיך המחשב לבצע את פעולת הקלט/פלט על המשתנים שנשארו בליסט.

דוגמא:

```
  PRINT 4,J,X,Q,K
4 FØRMAT (1X, I3, 2 F 5.1)
```

שלבי העבודה של המחשב יהיו במקרה זה כדלקמן.

1) המלה PRINT גורמת לכך שהמחשב יתחיל להדפיס מתחילת שורה חדשה.

2) המספר 4 פרושו שפקודה מספר 4 היא פקודת פורמט.

3) השדה 1X גורם לכך שהרוח בין השורה הנוכחית לשורה הקודמת, יהיה רוח יחיד.

4) שלושת המשתנים J, X, Q מודפסים לפי שלושת השדות I3, 2 F 5.1 .

5) כאשר המחשב מגיע למשתנה הרביעי K , הוא מוצא שהפורמט נגמר. לכן מבצע המחשב שתי פעולות. הוא חוזר לתחילת הפורמט ועובר לשורה חדשה.

כעת ממשיך המחשב בהדפסה. השדה 1X שבתחילת הפורמט אומר שוב להשאיר רוח יחיד. המשתנה K מודפס לפי השדה I3 וכאן נגמרת רשימת המשתנים, כלומר נגמרת פעולת ההדפסה.

הערה: מאחר והמשתנה הרביעי K מודפס לפי השדה הראשון שהוא I3 , חייב משתנה זה להיות שלם. פקודת הדפסה מהצורה PRINT 4, J, Q, X, U תגרום לשגיאה מאחר והמשתנה הרביעי U , שהוא ממשי, יודפס לפי השדה הראשון I3 , שהוא שלם.

דוגמא:

```
      READ 5, ((B (I, J), J = 1, 7), I = 1, 9)
5     FØRMAT (7F4. 1)
```

כאן כולל הליסט 63 משתנים ואילו הפורמט רק 7 שדות. שלבי הפעולה יהיו:

1) שבעת המשתנים הראשונים בליסט,
B(1, 1) B(1, 2) - - - B (1, 7)
יקראו בפורמט 7F4. 1 .

2) בנקודה זו יגיע המחשב לסוף הפורמט אולם הליסט עדיין נמשך. לכן יקרא המחשב כרטיס נוסף, יחזור לתחילת הפורמט ויקרא לתוך שבעת המשתנים הבאים B(2, 1) B(2, 2) - - - B(2, 7) לפי הפורמט 7F4. 1 .

3) תהליך זה יחזור לגבי כל קבוצה של שבעה משתנים, ומאחר שקיימות תשע קבוצות כאלה הרי שבסה"כ יקרא המחשב תשעה כרטיסים.

בסיכום: פקודת קריאה זו תגרום לכך שהמחשב יקרא תשעה כרטיסים ומכל כרטיס יקרא שבעה מספרים. מספרים אלה יוכנסו

למשתנים $B(I, 1)\ B(I, 2)\ \text{- - -}\ B(I, 7)$

כלומר לשורה ה- I של המשתנה B .

פקודה כזו מהווה לכן שיטה נוחה לקריאת מטריצה של מספרים לתוך משתנה בעל שני ממדים. מנקבים כל שורה של המטריצה על כרטיס נתונים, וקוראים בעזרת פקודה אחת כרטיסי נתונים שמספרם שוה למספר שורות המטריצה.

פרק 8

תכניות משנה (Sub Programs)

כשם שהשימוש במחשב בא כדי לחסוך עבודת חישוב, כך השימוש בתכניות משנה בא כדי לחסוך עבודה בכתיבת תכניות.

כאשר קטע מסוים בתכנית צריך להתבצע הרבה פעמים, שואף התכניתן לכתוב את הקטע רק פעם אחת ולחזור על ביצועו של אותו הקטע מספר פעמים כנדרש. דבר כזה אפשר לפעמים לבצע בעזרת הפקודה DØ , אולם יש מקרים בהם נוח יותר לכתוב את הקטע בתור תכנית משנה.

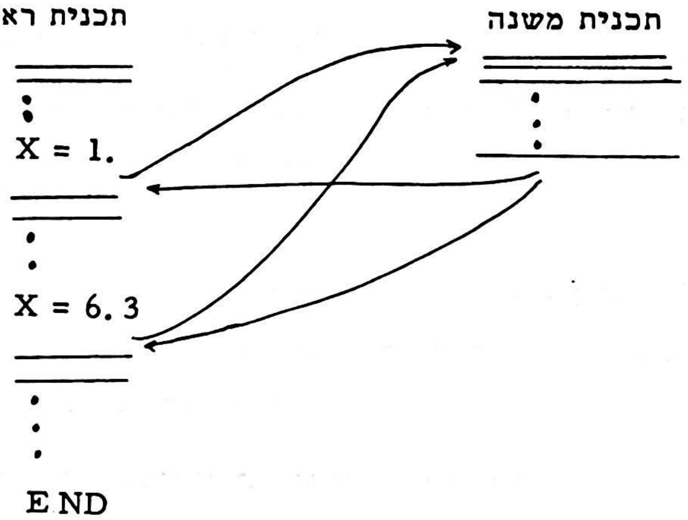
דוגמא: תכנית שבה רוצים לחשב סינוסים של שלוש זויות.

הפתרון הפשוט ביותר הוא לכתוב מערכת של פקודות לחישוב סינוס, ליצור שלושה העתקים של מערכת פקודות אלו ולהכניס כל העתק למקום בתכנית שבו נדרש חישוב של סינוס. ההעתקים יהיו שונים זה מזה רק בערכה של הזוית שאת הסינוס שלה יש לחשב.

שיטה זו אינה גרועה למקרה של שלוש זויות, אולם היא כמעט בלתי אפשרית למקרה של שלוש מאות זויות. במקרה כזה יהיה צורך ליצור שלוש מאות העתקים של הקטע המחשב סינוס, והעתקים אלה ינפחו את התכנית ללא צורך.

הפתרון הפשוט למקרה כזה הוא לכתוב קטע של תכנית המחשב סינוס של זוית שערכה נמצא במשתנה X . קטע זה יכתב בתור תכנית משנה ופרושו של דבר הוא שבכל פעם שיש לחשב סינוס, יוכנס ערכה של הזוית המתאימה לתוך המשתנה X , התכנית תקפוץ לתכנית המשנה, הסינוס יחושב ותכנית המשנה תחזור לתכנית הראשית לנקודה שממנה בוצעה הקפיצה.

הסכמה הבאה מראה תכנית שממנה קופצים פעמיים לתכנית משנה. בפעם הראשונה מחושב סינוס של 1. ובפעם השניה, סינוס של 6.3 של



תכנית המשנה חוזרת בכל פעם לנקודה ממנה בוצעה הקפיצה.

קיימים שני סוגים של תכניות משנה. תכניות משנה מהסוג הראשון נקראות סוב-רוטינות (Subroutines), ומהסוג השני, פונקציות (Functions).

<u>סוב-רוטינה.</u>

זוהי תכנית משנה שקופצים אליה בעזרת פקודה מיוחדת הנקראת CALL .

צורתה הכללית

CALL שם (פרמטר,פרמטר,- - -)

או

CALL שם

אחרי המלה CALL מופיע שמה של הסוברוטינה ואחריו יכולים להופיע פרמטרים, מופרדים ע"י פסיקים ומוקפים סוגרים.

המשמעות: המחשב יקפוץ לסוברוטינה ששמה צוין בפקודה, ימסור לה את ערכי הפרמטרים ולאחר שזו סיימה את פעולותיה, יחזור לתכנית הראשית לנקודה ממנה קפץ.

הפרמטרים שבפקודה CALL יכולים למלא שני תפקידים. הם יכולים למסור נתונים לסוברוטינה או להחזיר תוצאות שהסוברוטינה חישבה.

דוגמא: סוברוטינה לחישוב סינוס ששמה SINUS ולה שני פרמטרים. הראשון הוא ערכה של הזוית והשני הוא משתנה שלתוכו יש להכניס את התוצאה.

פקודת קפיצה לסוברוטינה זו יכולה להראות

CALL SINUS (31.2, Y)

המשמעות - על המחשב לקפוץ לסוברוטינה, היא תחשב את סינוס של 31.2 רדיאן ותכניס את התוצאה למשתנה Y . כאשר יחזור המחשב לתכנית הראשית, ימצא במשתנה Y המספר המתאים ואפשר יהיה להשתמש בו.

אפשר גם לכתוב CALL SINUS (X, Y) ואז יחושב סינוס של המספר הנמצא במשתנה X והתוצאה תוכנס ל- Y .

אפשר גם לכתוב CALL SINUS (X + 3.5, Z) ואז יחשב המחשב את הבטוי האריתמטי X + 3.5 . ימסור את התוצאה לסוברוטינה וזו תחשב סינוס של התוצאה ותכניס אותו למשתנה Z .

פרמטר המוסר נתונים לסוברוטינה יכול להיות בטוי אריתמטי אולם פרמטר המחזיר תוצאות מסוברוטינה חייב להיות שם של משתנה.

<u>דוגמא:</u> סוברוטינה בשם QE שתפקידה לפתור משואה רבועית. לסוברוטינה זו נרצה למסור את שלושת מקדמי המשואה וכן שני משתנים שלתוכם היא תכניס את שני הפתרונות. לכן יהיו לה חמשה פרמטרים.

פקודות קריאה לסוברוטינה זו יכולות להיות למשל:

```
CALL QE (A, B, C, X1, X2)
CALL QE (1., X- 35.  B, G, T1, T2)
CALL QE (A, B, - 15.3, U, W)
```

שלושת הפרמטרים הראשונים הם שלושת מקדמי המשואה (ולכן יכולים להיות בטויים אריתמטיים). שני הפרמטרים האחרונים הם שמות של משתנים ממשיים שלתוכם יש להכניס את התוצאות.

<u>מבנה הסוברוטינה</u>

הפקודה הראשונה בכל סוברוטינה חייבת להיות בעלת הצורה

SUBRØUTINE שם (פרמטר, פרמטר, - - -)

או SUBRØUTINE שם

אחריה באות פקודות רגילות ולבסוף הפקודה END .

בין הפקודות בסוברוטינה יכולה לבוא הפקודה RETURN . פקודה זו יכולה להופיע כמה פעמים בסוברוטינה ופרושה שנגמר ביצוע העבודה ועל המחשב לחזור לתכנית הראשית.

דוגמא: סוברוטינה בשם QE שתפקידה לפתור משואה רבועית.

```
SUBRØUTINE QE (Q1, Q2, Q3, T1, T2)
IF (Q1.NE.0) GØ TØ 10
T1 = -Q3/ Q2
T2 = T1
RETURN
```

```
10    DISQ = Q2 * * 2 - 4 * Q1 * Q3
      IF (DISQ) 30, 20, 40
20    T1 = - Q2 /(2* Q1)
      T2 = T1
      RETURN
30    PRINT 1000
1000  FØRMAT (14H CØMPLEX RØØTS)
      STØP
40    T1 = (- Q2 + DISQ * * .5) /(2 * A)
      T2 = (- Q2 = DISQ * * .5) /(2 * A)
      RETURN
      END
```

סוברוטינה זו נכתבה כך שהיא תקבל את מקדמי המשואה A, B, C בתור שלושת הפרמטרים הראשונים שלה (שסומנו כאן Q1 Q2 Q3) ותחזיר את התוצאות דרך שני הפרמטרים האחרונים T1, T2.

שלבי ביצוע החישוב הם

1) בדיקת המקדם הראשון Q1 . אם הוא אפס אזי המשואה איננה רבועית אלא ממעלה ראשונה והפתרון היחיד הוא Q3/Q2 - . פתרון זה מוכנס למשתנים T1 T2 וחוזרים (ע"י הפקודה RETURN) לתכנית הראשית.

2) מחושבת הדיסקרימיננטה Q2**2-4*A*C של המשואה.

3) אם הדיסקרימיננטה אפס, עוברת הסוברוטינה לפקודה 20, מחושב הפתרון וחוזרים לתכנית הראשית.

4) אם הדיסקרימיננטה שלילית, עוברת הסוברוטינה לפקודה 30, מודפסת הודעה האומרת ששרשי המשואה מרוכבים והעבודה מופסקת.

5) אם הדיסקרימיננטה חיובית, עוברת הסוברוטינה לפקודה 40, מחושבים שני הפתרונות וחוזרים לתכנית הראשית.

<u>הערות:</u>

1) בתוך סוברוטינה מותר להשתמש בפקודות READ PRINT.

2) מותר להשתמש בפקודה STØP בתוך סוברוטינה. במקרה זה לא תחזור הסוברוטינה לתכנית הראשית אלא העבודה תיפסק. משתמשים בפקודה זו כאשר מגלים שגיאה בתוך הסוברוטינה ומעונינים להפסיק את העבודה.

3) הפרמטרים של הסוברוטינה (במקרה שלנו (Q1, Q2, Q3, T1, T2) אינם הפרמטרים שעליהם יתבצע החישוב. כאשר כותבים את הסוברוטינה, אין לפרמטרים אלה ערכים מספריים. רק כאשר קוראים לסוברוטינה, כלומר רק בזמן בצוע הפקודה CALL, מקבלים פרמטרים אלה את הערכים שנותנת להם הפקודה CALL.

לדוגמא: הקריאה CALL QE(1., 2.5, X, Z, Z1) נותנת לפרמטר Q1 את הערך 1., לפרמטר Q2 את הערך 2.5 ולפרמטר Q3 את הערך שנמצא במשתנה X. הפרמטרים T1 T2 הופכים להיות זהים למשתנים Z Z1.
בקריאה הבאה לסוברוטינה QE, יכולים חמשת הפרמטרים שלה לקבל ערכים אחרים.

החישוב אינו מתבצע לכן על הפרמטרים של הסוברוטינה, אלא הוא מתבצע בכל פעם על פרמטרים אחרים (הפרמטרים שמוסרים לסוברוטינה בזמן הקריאה). לכן נקראים הפרמטרים של הסוברוטינה בשם פרמטרים פורמליים.

הפרמטרים המופיעים בפקודה CALL הם הפרמטרים האמיתיים שעליהם מתבצע החישוב. הם נקראים פרמטרים אקטואליים.

אפשר להעביר לסוברוטינה בתור פרמטרים, משתנים בעלי ממדים.

לדוגמא:

```
DIMENSIØN  A(100),  B(10, 3),  M(30), Z(200), W(8, 15)
:
CALL  SØS (A, B, 10, 3, M(15), Z)
:
END
SUBRØUTINE  SØS (X, Z, M, N, K, P)
DIMENSIØN  X(100),  Z(M, N),  P(50)
:
END
```

בדוגמא זו מועברים לסוברוטינה SØS שלושה משתנים בעלי ממדים.

A הוא וקטור בעל אורך 100. הוא מועבר לסוברוטינה בתור הפרמטר הראשון.

בתוך הסוברוטינה הוא נקרא X ושם הוא שוב מופיע בפקודה DIMENSIØN, כדי להודיע לסוברוטינה שהוא בעל ממד.

B היא מטריצה מספר 3 × 10. היא מועברת לסוברוטינה יחד עם המספרים 10,3 המציינים את ממדיה. ביחד הם מהווים את הפרמטרים השני, שלישי ורביעי בפקודה CALL .

בתוך הסוברוטינה נקראת המטריצה בשם Z והממדים נקראים M, N . Z מופיעה בפקודה DIMENSIØN עם הממדים M, N . זהו היוצא-מן-הכלל היחיד, לכלל האומר שהממדים בפקודה DIMENSIØN הם קבועים.

היוצא-מן-הכלל אומר שכאשר מופיעה הפקודה DIMENSIØN בתוך סוברוטינה, יכולים הממדים להיות משתנים, אולם משתנים אלה חייבים גם הם להיות פרמטרים של הסוברוטינה.

לדוגמא: תיתכן הקריאה הבאה

```
CALL SØS (A, W, 8, 15, M (15), Z)
```

כאן מעבירים לסוברוטינה את המטריצה W (עם הממדים שלה) במקום המטריצה B.

M הוא וקטור בעל אורך 30. כאשר רושמים M(15) בתוך הפקודה CALL פרוש הדבר שמעבירים לסוברוטינה את המספר היחיד הנמצא במקום ה-15 בוקטור M. כאן לא מעבירים משתנה בעל ממד אלא רק מספר יחיד. לכן מספר זה אינו מופיע בתוך הסוברוטינה בפקודה DIMENSIØN.

Z הוא וקטור בעל אורך 200. הוא מועבר לסוברוטינה בתור הפרמטר הששי. בתוך הסוברוטינה הוא נקרא P, ושם ארכו הוא רק 50.

קיים כלל האומר שכאשר משתנה בעל ממדים מועבר לסוברוטינה, חייב גדלו בתוך הסוברוטינה להיות שוה לגדלו בתכנית הראשית או קטן מזה.

לכן הוקטור Z יכול לעבור לסוברוטינה בתור וקטור באורך 50 או בכל אורך שהוא שלא יעלה על 200.

בתוך סוברוטינה יכולים להופיע משתנים בעלי ממדים, שאינם פרמטרים של הסוברוטינה.

לגבי משתנים אלה קיימים כל הכללים הרגילים.

לדוגמא:

```
SUBRØUTINE ØLD (P, Y, M)
DIMENSIØN P (13), X (24)
   .
   .
   .
END
```

X הוא משתנה בעל ממדים, אולם אינו מופיע כפרמטר של הסוברוטינה. זהו משתנה פנימי של הסוברוטינה שאינו מועבר לה מהתכנית הראשית.

<u>פונקציות</u>

פונקציה היא תכנית משנה שהקריאה אליה איננה מתבצעת ע"י פקודה מיוחדת אלא ע"י הזכרת שמה במשפט יחוס אריתמטי. התוצאה שמחשבת הפונקציה חוזרת אל התכנית הראשית כשהיא קשורה לשמה של הפונקציה.

תוצאת החישוב של הפונקציה חייבת להיות מספר יחיד. תוצאת החישוב של סוברוטינה יכלה להיות כמה מספרים.

דוגמא: תהי FNCT פונקציה בעלת שני פרמטרים.

הפקודה Z = FNCT (X, Y) היא משפט יחוס אריתמטי. המחשב מבצע משפט זה בשלבים הבאים:

1) הוא קורא לפונקציה FNCT . נותן לה את הפרמטרים X Y , והפונקציה מחשבת מה שמחשבת.

2) בחזרה מהפונקציה נמצא המספר, שהוא תוצאת החישוב של הפונקציה, בתוך משתנה ששמו כשם הפונקציה FNCT .

3) המחשב מחשב את אגף ימין. במקרה זה ערכו של אגף ימין הוא הערך שחישבה הפונקציה (או הערך של המשתנה FNCT). ערך זה מוכנס למשתנה Z המופיע באגף שמאל.

<u>דוגמא:</u> Z = FNCT (30, X + 15.) - 134.18 .

שלבי הביצוע של המחשב יהיו:

1) חישוב הפרמטרים של הפונקציה. ערכו של הפרמטר הראשון הוא 30, ושל השני .X + 15.

2) קריאה לפונקציה.

3) בגמר החישוב של הפונקציה נמצאת תוצאת החישוב בתוך המשתנה FNCT.

4) חישוב אגף ימין FNCT - 134.18 והכנסת הערך שלו למשתנה Z .

<u>הערות:</u>

1) תוצאת החישוב של הפונקציה מוחזרת בעזרת שמה של הפונקציה ולא בעזרת פרמטר. לכן כל הפרמטרים של פונקציה הם פרמטרים המוסרים נתונים לפונקציה ולכן כולם יכולים להיות בטויים אריתמטיים. בטויים אלה מחושבים לפני הקריאה לפונקציה.

2) הקריאה לפונקציה מהווה חלק מבטוי אריתמטי (ראה הגדרת בטוי אריתמטי בפרק 2). לאחר הקריאה לפונקציה מתיחס המחשב אל שמה של הפונקציה כאל משתנה רגיל המשתתף בבטוי האריתמטי.

3) פרמטר של פונקציה הוא בטוי אריתמטי, ובטוי אריתמטי יכול להכיל קריאות לפונקציות. לכן יכול פרמטר של פונקציה להכיל בעצמו פניות לפונקציות אחרות.

<u>דוגמא:</u> תהיינה F1 F2 פונקציות.

המשפט Y = F1 (J, F2 (130), K) הוא משפט יחוס הקורא לפונקציה F1 כאשר הפרמטר השני שלה הוא קריאה לפונקציה F2 .

<u>צורת כתיבת הפונקציה</u>

המשפט הראשון הוא

FUNCTIØN שם (פרמטר, פרמטר, - - -)

אח"כ באות פקודות FØRTRAN רגילות, שביניהן לפחות

פקודה אחת בעלת הצורה --- = שם.

בסוף, הפקודה END .

החזרה לתכנית הראשית היא ע"י RETURN.

דוגמא: פונקציה SUM המקבלת בתור פרמטר משתנה בעל ממד ומסכמת את אברי המשתנה.

```
   FUNCTIØN SUM (A, M)
   DIMENSIØN A (M)
   SUM = 0
   DØ 10 I = 1, M
10 SUM = SUM + A (I)
   RETURN
   END
```

A מופיע בפקודה DIMENSIØN עם ממד משתנה.
שמה של הפונקציה משמש גם כמשתנה שלתוכו מוכנסת התוצאה. מסיבה זו יש לבחור לפונקציה שם שיתאים לסוג התוצאה המתקבלת. אם התוצאה היא מספר שלם, חייב שמה של הפונקציה להתחיל באחת מהאותיות IJKLMN:

קריאה לפונקציה SUM יכולה למשל להיות כבדוגמא הבאה:

```
DIMENSIØN X (16), Z (89)
 .
 .
Y = SUM (X, 16) + SUM (Z, 89) + 1.
 .
 .
 .
END
```

לתוך המשתנה Y נכנס סכום 16 אברי הוקטור X , 89 אברי הוקטור Z ועוד המספר 1.

כל תכניות המשנה של תכנית ראשית חייבות להופיע אחרי כרטיס END של התכנית הראשית, ואין חשיבות לסדר הופעתן.

הקשר בין תכנית ראשית לתכנית משנה נעשה באמצעות הפרמטרים. משתנה המופיע בתכנית ראשית ואינו משמש פרמטר בקריאה לתכנית משנה, אינו מוכר לתכנית המשנה והיא איננה יכולה להשתמש בו או לשנות אותו.

לדוגמא:

```
      .
      .
      .
      I = 5
      CALL AB (X, Y)
      .
      .
      .
      END
      SUBRØUTINE AB (A, B)
      .
      .
      I = 3
      .
      .
      END
```

המשתנה I מופיע בתכנית הראשית וגם בסוברוטינה. הוא אינו מופיע בתור פרמטר של הסוברוטינה ולכן הוא משתנה פנימי שלה. במלים אחרות, המשתנה I בתכנית הראשית והמשתנה I בסוברוטינה הם משתנים שונים ושנוי של אחד מהם אינו משפיע על השני.

CØMMØN

המצב המתואר בסעיף הקודם יכול להשתנות אם המשתנה I יוגדר ב-CØMMØN.

הצורה הכללית של הפקודה היא COMMON list

פקודה זו מאפשרת לתכנית משנה לקבל משתנים מהתכנית הראשית שלא באמצעות פרמטרים.

דוגמא:

```
COMMON I, A, C
  .
  .
CALL GEN (X, Y, Z)
  .
  .
END
SUBROUTINE GEN (X, Y, Z)
COMMON I, B
  .
  .
END
```

המשתנה I בתכנית הראשית יהיה זהה למשתנה I בסוברוטינה מכיון ששניהם נמצאים במקום הראשון בפקודה COMMON.

המשתנה A בתכנית הראשית יהיה זהה למשתנה B בסוברוטינה מכיון ששניהם נמצאים במקום השני בפקודה COMMON.

המשתנה C בתכנית הראשית לא יהיה מוכר לסוברוטינה כי ה- COMMON שלה מכיל רק שני משתנים.

קיימים הכללים הבאים.

1) ארכו של ה- COMMON בכל תכנית משנה יכול להיות קטן או שוה לארכו בתכנית הראשית.

2) המשתנים המותאמים באמצעות ה- COMMON יכולים להיות שונים בשמותיהם אולם חייבים להיות שוים באופים.

3) משתנה יכול לקבל ממד בפקודה COMMON אולם אז אסור לו להופיע גם בפקודה DIMENSION.

<u>דוגמא:</u> CØMMØN A, B (200)

למשתנה B אסור להופיע בפקודה DIMENSIØN.

<u>פונקציות בנויות</u>

התכונה החשובה של פונקציה היא שתוצאת החישוב מוחזרת באמצעות שמה של הפונקציה ולא באמצעות פרמטר.

תכונה זו אפשר לנצל בצורה הבאה.

יכול תכניתן לכתוב פונקציה לחישוב סינוס. הפונקציה תיקרא SIN ויהיה לה פרמטר אחד, המספר שאת הסינוס שלו יש לחשב.

אפשר להעתיק את כרטיסי הפונקציה (החל בכרטיס FUNCTIØN וכלה בכרטיס END) ולתת את ההעתק לאדם אחר.

אותו אדם יכול לקרוא לפונקציה ולהשתמש בה למרות שאולי אינו יודע איך היא כתובה ובאיזה שיטה היא מחשבת.

מאחר וחישוב של סינוס הוא חישוב שימושי, הרי הגיוני יותר יהיה להכניס את הפונקציה לזכרון של המחשב כך שכל תכניתן יוכל להשתמש בה ע"י עצם הזכרת שמה במשפט יחוס אריתמטי.

דבר כזה ניתן לבצוע והוא בוצע לגבי כמה פונקציות שמושיות. בפונקציות אלה אפשר להשתמש מבלי שיהיה צורך לכתוב אותן מחדש. הן מצויות תמיד בתוך המחשב או, במלים אחרות, הן בנויות במחשב ואפשר תמיד להשתמש בהן. פונקציות כאלה נקראות פונקציות בנויות. כדי להשתמש בהן צריך לדעת רק מה שמן ומה הפרמטר (או פרמטרים) שלהן.

הטבלה להלן מתארת את הפונקציות הבנויות החשובות ביותר.

שם	תאור	
EXP (X)	e^X	אקספוננט
ALØG (X)	ln X	לוג טבעי
ALØG 10 (X)	lg X	לוג על בסיס 10
SIN (X)	sin X	סינוס
CØS (X)	cos X	קוסינוס
ABS (X)	$\|X\|$	ערך מוחלט
IABS (I)		ערך מוחלט של משתנה שלם
FLØAT (I)		הפיכת משתנה שלם לממשי
SQRT (X)	$\sqrt{X}$	שרש

הפונקציה IABS מקבלת בתור פרמטר, משתנה שלם, ונותנת בתור תוצאה, מספר שלם.

הפונקציה FLØAT מקבלת בתור פרמטר, משתנה שלם, ונותנת בתור תוצאה, מספר ממשי.

פרט לכך מקבלות כל הפונקציות שלמעלה, פרמטר ממשי ונותנות תוצאה ממשית.

קיימות עוד פונקציות בנויות שהן פחות שימושיות והן מפורטות בנספח מס' 1.

לגבי תכניות משנה קיים הכלל החשוב הבא: כל תכנית משנה יכולה לקרוא לתכניות משנה אחרות, אולם אסור לתכנית משנה לקרוא לעצמה, וכן אסורה שרשרת של קריאות (כמו A קוראת ל- B , B קוראת ל- C , C קוראת ל- A).

פרק 9

מבנה של עבודה (Job)

בפרק זה יתואר המבנה של עבודות (Jobs) המוגשות למחשב CDC 6400 .

עבודות המוגשות למחשבים אחרים יכולות להיות שונות לחלוטין במבניהן ופרק זה אינו מתיחס אליהן.

קיים הבדל חשוב בין כתיבת תכנית למחשב ובין הגשת עבודה למחשב. עבודה (Job) למחשב, כוללת בתוכה תכנית, נתונים לתכנית וכן הוראות למחשב לגבי בצוע התכנית. לכן המושג של עבודה (Job) למחשב הוא כללי יותר ממושג התכנית.

במחשב CDC 6400 מורכבת כל עבודה מקטעים (Records) המופרדים זה מזה ע"י כרטיס מיוחד. כרטיס זה נקרא END - ØF - RECØRD והוא בעל ניקוב מיוחד (ניקוב 9-8-7 בעמודה 1). סוף העבודה מצוין גם הוא ע"י כרטיס מיוחד הנקרא END - ØF - INFØRMATIØN והוא בעל ניקוב 9-8-7-6 בעמודה 1.

הקטע (Record) הראשון בעבודה מכיל כרטיסי הוראות (Control Cards). כרטיסים אלה מכילים אינפורמציה האומרת למחשב מה יבוא בקטעים הבאים של העבודה. הקטעים הבאים עלולים להכיל תכנית (כתובה FORTRAN או בשפה אחרת), או מספר תכניות, והם עלולים להכיל נתונים לאותן תכניות.

להלן יתוארו כרטיסי ההוראות של עבודה המכילה תכנית FORTRAN.

כרטיס ראשון:

עמודות 1-10 שם התכניתן .

עמודות 11-20 שם העבודה .

שמות אלה יודפסו אח"כ יחד עם תוצאות החישוב ויאפשרו לתכניתן לזהות את עבודתו בין כל העבודות הנפלטות מהמחשב.

כרטיס שני:

עמודות 1-6 מכילות RUN (S).

כרטיס זה אומר למחשב שהקטע השני של העבודה מכיל תכנית F0RTRAN.

כרטיס שלישי:

עמודות 1-4 מכילות LGØ..

כרטיס זה אומר למחשב שעליו להפעיל את התכנית אם אין בה שגיאות לוגיות או תחביריות.

כדי להבין את משמעותם של שני כרטיסים אלה, יש להבין את הדרך בה מבצע המחשב תכנית F0R.TRAN . באופן כללי כולל הביצוע שני שלבים. בשלב הראשון עובר המחשב על כרטיסי התכנית ומנסה לגלות בהם שגיאות לוגיות או תחביריות. אם לא נתגלתה כל שגיאה, חוזר המחשב לתחילת התכנית ומתחיל לבצע אותה, פקודה אחר פקודה. אם נתגלו שגיאות, לא יתחיל המחשב לבצע את התכנית.

המעבר הראשון מתבצע ע"י קריאת כרטיס RUN (S) . אם לא נתגלו שגיאות, מוכן המחשב לבצע את התכנית במעבר השני, אולם הוא יעשה זאת רק אם יהיה כרטיס LGØ. שיאמר לו שהתכניתן מעונין בהפעלת התכנית.

במידה ויהיה רק כרטיס RUN (S) , יעבור המחשב על התכנית פעם אחת וינסה לגלות בה שגיאות. הוא לא יבצע אותה.

קיימים שני סוגי שגיאות, לוגיות ותחביריות. שגיאות לוגיות הן שגיאות העלולות לגרום לכך שתכנית לא תעבוד למרות שכל הפקודות שלה הן נכונות מבחינה תחבירית.

לדוגמא: הפקודה X = 3. נכונה מבחינה תחבירית, אולם פקודה זו עלולה לגרום לשגיאה לוגית אם היא נמצאת בהקשר לא נכון. הקטע הבא מכיל פקודה זו ויש בו שגיאה לוגית.

```
 GØ TØ 50
 X = 3.
```

השגיאה היא שהמחשב לא יוכל להגיע לפקודה X = 3. כאשר הוא יבצע את התכנית. הוא יבצע את הפקודות אחת אחרי השניה, וכשיגיע לפקודה שלפני X = 3. , יקפוץ משם לפקודה מספר 50 וכך לא יגיע אף פעם לפקודה X = 3. . זו שגיאה לוגית שהמחשב יכול לגלותה למרות שאינו יודע מה עושה התכנית ומה תפקידו של המשתנה X .

הקטע הבא

```
   GØ TØ 50
4  X = 3.
```

אינו מכיל שגיאה לוגית, מכיון שהמחשב יכול להגיע לפקודה X = 3. אם קיימת באיזשהו מקום הפקודה GØ TØ 4.

שגיאות תחביריות הן עבירות על כללי הדקדוק של השפה. לדוגמא: הפקודה READ 1 X מכילה שגיאה תחבירית מכיון שחסר בה הפסיק בין מספר הפורמט והליסט.

לאחר כרטיסי הקטע הראשון מופיע כרטיס
END - ØF - RECØRD . אחריו באה התכנית המתחילה
בכרטיס PRØGRAM ונגמרת בכרטיס END , אחריה
באות תכניות המשנה (אם יש) המתחילות כל אחת בכרטיס
SUBRØUTINE או FUNCTIØN ונגמרות בכרטיס
END .

לאחר כרטיס END של תכנית המשנה האחרונה יש שתי
אפשרויות.

1) אם יש כרטיסי נתונים, יבוא כרטיס END - ØF - RECØRD ,
אחריו כרטיסי הנתונים ואחריו כרטיס
END - ØF - INFØRMATIØN.

2) אם אין כרטיסי נתונים, יבוא כרטיס
END - ØF - INFØRMATIØN.

כללים אלה מסוכמים בסכמות הבאות.

עבודה הכוללת תכנית עם נתונים:

```
שם --
RUN (S)
LGØ.
end-of-record
        PRØGRAM --
        :
        .
        END
        SUBRØUTINE --
        :
        .
        END
```

```
          FUNCTIØN - -
          .
          .
          .
          END
end-of-record
   כרטיסי נתונים
end-of-information
```

עבודה הכוללת תכנית בלי נתונים:

```
שם- -
RUN (S)
LGØ.
end-of-record
          PRØGRAM - -
          .
          .
          .
          END
          FUNCTIØN - -
          .
          .
          .
          END
          FUNCTIØN
          .
          .
          .
          END
end-of-information
```

יש לשים לב לכך שכרטיסי ההוראות מתחילים מעמודה 1. כרטיסי ההוראות אינם כרטיסי FØRTRAN , הם אינם מהווים חלק מהתכנית אלא משמשים הוראות למחשב לטיפול בתכנית.

בדומה גם כרטיסי הנתונים אינם כרטיסי FØRTRAN .
הם אינם חלק מהתכנית אלא הם נתונים שהתכנית קוראת. לכן
גם כרטיסים אלה לא חייבים להתחיל מעמודה מסוימת אלא המבנה
שלהם נקבע ע"י הפורמט שלפיו הם נקראים.

אם כרטיס נתונים נקרא ע"י הפורמט (3I5 , 14 X) אזי
הכרטיס צריך להתחיל מעמודה 15 ולהגמר בעמודה 29.
ניקובים הנמצאים על הכרטיס לפני עמודה 15 או אחרי 29,
אינם מהוים שגיאה אולם המחשב לא יקרא אותם.

פרק 10

גדלים אלפמריים

כל הקבועים והמשתנים שתוארו עד עכשיו היו מספריים. על גדלים אלה אפשר היה לבצע פעולות אריתמטיות או להשוות ביניהם ממש כשם שמשווים מספרים.

המקרה היחיד בו הופיעו גדלים לא מספריים היה פורמט H . פורמט זה משמש להדפסת כותרות ולכן אפשר להדפיס באמצעותו כל קרקטר שהוא ולא דוקא מספרים.

כדי לאפשר שימוש נוח בגדלים לא מספריים, קיימים ב- FORTRAN קבועים ומשתנים אלפמריים.

גדלים אלה אינם מכילים מספרים ולכן אין לבצע עליהם פעולות אריתמטיות. הם מכילים קרקטרים כלשהם (אותיות, ספרות או סימנים מיוחדים) ואפשר רק לקרוא אותם מכרטיסי נתונים, להדפיס אותם או להשוות ביניהם כפי שמשווים מלים. (אין משמעות לשאלה האם גודל אלפמרי אחד גדול או קטן מהשני, אלא רק לשאלה האם גודל אלפמרי אחד שוה לשני או שונה ממנו).

הגדלים האלפמריים מתחלקים לקבועים ולמשתנים.

<u>הגדרה:</u> קבוע אלפמרי הוא בעל הצורה ($n H a_1 a_2 \ldots a_n$) כאשר n מספר בין אחד לעשר (במחשבים אחרים יכול ערכו המקסימלי של n להיות שונה מעשר), $a_1 \ldots a_n$ קרקטרים כלשהם.

<u>דוגמאות:</u> 1) (3HABC) זהו קבוע אלפמרי המכיל את שלושת הקרקטרים ABC .

2) (4H, b$3) זהו קבוע אלפמרי המכיל את ארבעת הקרקטרים 3 $ b ,

הקרקטר b נספר כמו כל קרקטר אחר וזהו יוצא מן הכלל לחקי הכתיבה של השפה (האומרים שבלנק יכול להופיע בכל מקום ואינו נחשב).

3) (2H63) זהו קבוע אלפמרי המכיל את שני הקרקטרים 3 6. הוא מכיל רק ספרות ובכל זאת איננו קבוע מספרי. ערכו של הקבוע איננו המספר 63 אלא שני הקרקטרים 3 6.

<u>הגדרה:</u> משתנה אלפמרי הוא משתנה המכיל קרקטרים ולא גודל מספרי.

שמו של משתנה אלפמרי כפוף לכללים הרגילים אולם הוא אינו חייב להתחיל באות מסוימת. גם המשתנה I וגם המשתנה Y יכולים להיות משתנים אלפמריים (אם הם מכילים קרקטרים). המחשב אינו יודע אם המשתנה I הוא משתנה שלם או אלפמרי. התכניתן חייב לדעת, לגבי כל משתנה, אם הוא מכיל מספר או קרקטרים. אם משתנה מסוים הוא אלפמרי, אין לבצע אתו פעולות חשבון. במידה ובכל זאת מכניסים משתנה אלפמרי בתוך בטוי אריתמטי, המחשב לא יצין שום שגיאה אולם התוצאה לא תהיה נכונה.

<u>משפט יחוס אלפמרי</u>

זהו משפט מהצורה $a = (n\,H\,a_1\,a_2 \ldots a_n)$.

כאשר אגף שמאל הוא שם של משתנה (ממשי או שלם) ואגף ימין הוא קבוע אלפמרי.

<u>דוגמאות</u> 1) J = (3 H 1 X,)

משפט זה מכניס את שלושת הקרקטרים , X 1 למשתנה J
והחל מכאן יהיה J משתנה אלפמרי.

במהלך התכנית אפשר להפוך את J שוב למשתנה מספרי,
וזאת למשל ע"י משפט מהצורה J = 7 .

2) X = (1 H b)

משפט זה מכניס את הקרקטר היחיד b (בלנק) למשתנה X.

3) M11 = (2 H 11)

משפט זה מכניס את שני הקרקטרים 11 למשתנה M11 .
משתנה זה הופך להיות אלפמרי למרות שהוא מכיל רק קרקטרים
ספרתיים. לאחר בצוע הפקודה שלמעלה לא מכיל המשתנה
M 11 את המספר 11 או כל מספר אחר ואין לבצע עליו פעולות
חשבון.

במחשב 6400 CDC יכול כל משתנה אלפמרי להכיל עד
עשרה קרקטרים, במחשבים אחרים עלול המספר להיות שונה.

אם מכניסים למשתנה אלפמרי פחות מעשרה קרקטרים,
ישלים המחשב את מספר הקרקטרים לעשרה, ע"י הוספת קרקטרים
ריקים (בלנק).

דוגמא: 1) K = (1 H K) המשתנה K יכיל את הקרקטר
K ועוד תשעה בלנקים.

2) H = (1 H b) המשתנה H יכיל את הקרקטר b (בלנק)
ועוד תשעה בלנקים (כלומר בסה"כ עשרה בלנקים).

<u>הפקודה DATA</u>

פקודה זו משמשת להכנסת קבועים (מספריים או אלפמריים)
לתוך משתנים. הפקודה מתבצעת לפני תחילת בצוע התכנית.

הצורה הכללית   DATA list / clist /.

כאשר list היא רשימה של משתנים, clist – רשימה של קבועים.

דוגמאות: 1) DATA A, B, C / 1. , (2HCC), - 30.5 /

פקודה זו תכניס את הקבוע 1. למשתנה A , את הקרקטרים CC למשתנה B (שיהפוך להיות משתנה אלפמרי) ואת הקבוע 30.5 - למשתנה C .

2)

```
DIMENSIØN A (4)
DATA A/1., 3.5, - .17, 0./
```

A מוגדר כמשתנה בעל אורך 4. הפקודה DATA מכניסה ערכים מספריים לארבעת המקומות במשתנה A .

3)

```
DIMENSIØN A (10)
DATA (A(I), I = 1, 4) /1., 2., 3., 4.5/
```

כאן A הוא בעל עשרה מקומות והפקודה DATA מכניסה ערכים רק לארבעת המקומות הראשונים.

כאשר מופיע בפקודה DATA משתנה בעל ממד, חייב המשתנה להופיע בפקודה DIMENSIØN לפני שהוא מופיע בפקודה DATA . פרט להגבלה זו יכולה הפקודה DATA להופיע בכל מקום בתכנית אולם היא מבוצעת רק פעם אחת, בתחילת בצוע התכנית.

דוגמא:

```
   .
   .
 ______
 ______
20 ______
   DATA ...
50 ______
 ______
   .
   .
```

כאשר המחשב יגיע לפקודה מספר 20 הוא ידלג על הפקודה DATA (שבוצעה בתחילת בצוע התכנית) וימשיך בפקודה מספר 50.

משתנה שקבל ערך בפקודה DATA יכול אח"כ לקבל ערך אחר (ע"י משפט יחוס למשל).

אסור להכניס משתנה גם לפקודה CØMMØN וגם לפקודה DATA.

<u>פורמט A</u>

על מנת להדפיס משתנה אלפמרי, או לקרוא קרקטרים מכרטיס לתוך משתנה אלפמרי, קיים בפקודת הפורמט שדה מיוחד הנקרא שדה A.

צורתו הכללית Aw, כאשר w - מספר בין אחד לעשר (במחשבים אחרים שונה בדרך כלל ערכו המקסימלי של w).

המשמעות: לגבי קריאה - המחשב יקרא w קרקטרים מכרטיס הנתונים ויכניס אותם לתוך המשתנה המופיע בפקודה READ. משתנה זה יהפוך להיות אלפמרי. אם w קטן מעשר ישלים המחשב קרקטרים ריקים (בלנקים) בתוך המשתנה.

לגבי הדפסה - w קרקטרים יודפסו בשדה המתאים על הניר, החל מצד שמאל של השדה.

דוגמאות: (1

```
READ 12, A3
12 FØRMAT (A4)
```

אם כרטיס הנתונים הוא |1 2 3 X

אזי יכנסו למשתנה A3 ארבעת הקרקטרים 1 2 3 X
ואחריהם ששה בלנקים.

(2

```
PRINT 2, A3
2 FØRMAT ( 1 X, A 2)
```

ההדפסה תהיה - 12

המשתנה A3 מכיל ארבעה קרקטרים אולם מותר להדפיס רק חלק מהם. בדוגמא 2 הודפסו שני השמאליים.

(3

```
PRINT 3, A3
3 FØRMAT (1 X, A 5)
```

ההדפסה תהיה 1 2 3 X b .

משתנים וקבועים אלפמריים יכולים להופיע בתוך בטויים לוגיים, אולם רק עם פעולות היחוס .EQ. .NE. .

דוגמא: (1

```
IF (J X.EQ.(3 H A B C)) X = 1.5
```

(2

```
A = (1 H A)
READ 2, B
2 FØRMAT (A 1)
IF (A.NE.B) GØ TØ 7
.
.
.
```

גדלים אלפמריים משמשים לקריאת כותרות מכרטיסים, להדפסת כותרות, לציור גרפים ולהשואות לא מספריות.

הדוגמאות שלהלן מראות כיצד ניתן להשתמש בגדלים אלפמריים למטרות אלה.

1) כרטיסי נתונים מנוקבים בפורמט הבא:

בעמודות 1-4 אחת מהמלים SQRT , SIN, LØG

בעמודות 5-14 מספר ממשי בעל שלוש ספרות אחרי הפסיק.

יש לקרוא את כרטיסי הנתונים ולחשב שורש או סינוס או לוגריתמוס של המספר המופיע בכל כרטיס, לפי המלה המתאימה המנוקבת בו:

התכנית היא:

```
10  READ 100, W, U
100 FØRMAT (A4, F10. 3)
    IF ( W. EQ. (3HEND)) STØP
    IF (W. EQ. (4HSQRT)) X = SQRT (U)
    IF (W. E Q. (3HSIN)) X = SIN (U)
    PRINT 200, W,U., X
200 FØRMAT (1X, A4, F11. 3, F8. 4)
    GØ TØ 10
    END
```

לאחר כרטיסי הנתונים מוסיפים כרטיס ובו מנוקבת המלה END בעמודות 1-3.

התכנית קוראת כל כרטיס לתוך המשתנים W (אלפמרי) U(ממשי).

אם לתוך W נקראה המלה END , התכנית עוצרת.

אחרת בודקת התכנית אם נמצאת ב- W אחת מהמלים SQRT SIN LØG ומחשבת את ערכו של X בהתאם.

הערכים של W U X מודפסים והפקודה האחרונה

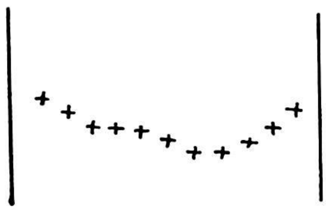
GØ TØ 10 פרושה לחזור ולקרוא את כרטיס הנתונים הבא.

2) הדפסת גרף.

לעתים נוח להציג תוצאות של חשובים מסוימים לא בצורה הרגילה של טבלה אלא בצורה גרפית.

אפשר להדפיס גרף בעזרת המחשב ע"י הדפסת קרקטרים על הניר, כך שהרושם הכללי יהיה של קו.

בדוגמא נראה קו גלי שהתקבל ע"י הדפסת הקרקטר + מספר פעמים



הקו אינו רציף כמובן אולם כאשר מסתכלים עליו ממרחק מספיק, מקבלים התרשמות של קו ולא של אוסף קרקטרים.

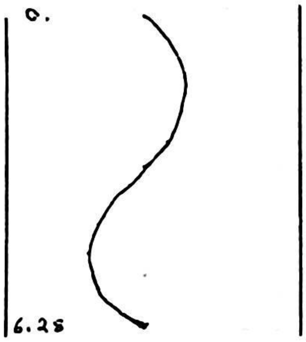
כאשר רוצים לצייר שני גרפים על אותו דף, אפשר להבחין ביניהם על ידי שימוש בקרקטרים שונים.

אפשר גם לצייר צירים ע"י שמוש בקרקטר - בשביל ציר X ובקרקטר I בשביל ציר Y .

התכנית הבאה מצירת גרף של סינוס שלם לאורך הדף.

```
      DIMENSIØN  LIN (101)
      DØ 10 I = 1, 101
10    LIN (I) = (1Hb)
      DØ 100 I = 1, 629
      X = I -1
      X = X/100
      J = SIN (X)* 50 + 51.5
```

```
      LIN (J) = (1H*)
      PRINT 1000, X, LIN
1000  FØRMAT (F6.2, 3X, 101A1)
 100  LIN (J) = (1Hb)
      STØP
      END
```



הסבר:

הגרף צריך להראות כבציור.

ההדפסה של הגרף נעשית שורה שורה כאשר בכל שורה מדפיסים את הערך של הזוית (בין אפס לשני רדיאן) ומדפיסים נקודה אחת של הגרף.

פקודת ההדפסה היא PRINT 1000, X, LIN

X הוא ערכה של הזוית. שלושת הפקודות

```
DØ 100 I = 1, 629
X = I - 1
X = X / 100
```

גורמות לכך שהמשתנה X יקבל ערכים מאפס עד 6.28 בקפיצות של מאית.

LIN הוא וקטור בעל 101 מקומות המודפס בפורמט (101 A1).

הרעיון בתכנית הוא להכניס לאחד המקומות בוקטור LIN את הקבוע האלפמרי (1H*), ולשאר המקומות להכניס בלנקים. בצורה זו נותנת הדפסה של LIN, שורה שבה מופיע כוכב אחד.

כל מה שנותר לקבוע הוא מקומו של הכוכב בוקטור LIN לכל שורה ושורה בהדפסה.

מקום זה נקבע ע"י הפקודה

J = SIN (X)* 50 + 51.5

הסבר: הגודל SIN (X) מקבל ערכים בין -1. ל +1.

הגודל SIN (X) * 50 מקבל ערכים בין -50. ל +50.

הגודל SIN (X) * 50. + 51. מקבל ערכים בין +1. ל +101.

כאשר מעבירים גודל זה למשתנה J יקבל J ערך בין 1 ל 101.

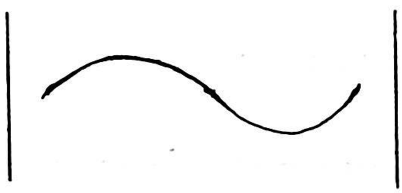
לפנינו טרנספורמציה לינארית של Sin X, טרנספורמציה המעבירה כל ערך של הסינוס לערך בין 1 ל 101.

הפקודה LIN (J) = (1H *) מכניסה כוכב לוקטור LIN במקום המתאים לסינוס שחושב (מתאים לאחר הטרנספורמציה).

בסה"כ מקבלים הדפסה של 629 שורות שבכל אחת מהן מספר (שהוא ערכה של הזוית המתאימה) וכוכב הנמצא במקום המתאים לסינוס שחושב.

<u>תרגילים:</u>

1) ציור גרף של סינוס לרוחב הניר כלומר:



כאן יש בכל שורה של ההדפסה בדרך כלל שני כוכבים (אולם יש מקרים של כוכב אחד ומקרה של שלושה כוכבים).

2) ציור מעגל.

לתרגיל זה יש לקחת בחשבון שהמחשב מדפיס שש שורות לאינטש

ובתוך כל שורה נכנסים עשרה קרקטרים לאינטש. לכן היחס בין
מספר השורות (N) שהתכנית תדפיס לבין מספר הקרקטרים (L)
בכל שורה צריך להיות $\frac{N}{L} = \frac{5}{3}$. כל יחס אחר יתן
אליפסה במקום מעגל.

פרק 11

סרטים מגנטיים

מאחר ולמחשב זכרון מוגבל, הרי עלול להיוצר מצב שבו גודל הזכרון לא יספיק כדי לאחסן את כל הנתונים הדרושים, במקרה כזה אפשר להשתמש בסרטים מגנטיים כדי לאחסן עליהם נתונים.

במחשב מורכבים סרטים מגנטיים כמו ברשמקול וההבדל העקרוני היחידי הוא שברשמקול נעשית הכתיבה על הסרט באופן רציף ובמחשב נרשמים הנתונים קרקטר אחרי קרקטר. הקרקטרים ארוזים בבלוקים שארכם ניתן לשנוי ובין בלוק לבלוק קים רוח ריק על הסרט.

בשיטת עבודה אחת רושמים את הנתונים על הסרט בחלקה הראשון של העבודה, אח"כ מגלגלים את הסרט לתחילתו והחלק השני של העבודה יכול לקרוא מהסרט את הנתונים. שיטה זו משתמשת בסרט בתור טיוטא (Scratch). בשיטת עבודה שניה רושמים את הנתונים על הסרט בעבודה אחת ואח"כ שומרים על הסרט ואפשר להשתמש בו לעבודות אחרות. עבודות אלה יכולות לקרוא את הנתונים שעל הסרט וכן יכולות להוסיף עליו עוד נתונים. בשיטה זו משתמשים בסרט בתור אוסף אישי של נתונים והסרט חיב להיות שמור לאדם אחד.

אפשר לכתוב על סרט (ולקרוא ממנו) אינפורמציה משני סוגים. סוג ראשון הוא תוכן של משתנים בזכרון. סוג שני הוא כותרות.

כדי להשתמש בסרט יש קודם כל לקבוע על איזו יחידה יהיה הסרט מורכב. לכל מחשב יש מספר יחידות (או כוננים) עליהן אפשר להרכיב סרטים והן ממוספרות במספרים סדוריים המשתנים ממחשב למחשב.

ארבע הפקודות הראשונות שנראה הן:

```
READ(k, n) list
WRITE(k, n) list
BACKSPACE k
REWIND k
```

כאשר k - מספר היחידה (הכונן) עליה מורכב הסרט,
n - מספר של פקודת פורמט,
list - רשימה של שמות משתנים מופרדים ע"י פסיקים.

קריאה וכתיבה על סרט דומות לקריאה והדפסה רגילות וקימים שני הבדלים חשובים.

1) לא קים המושג של שורות ולכן בכתיבה על סרט מהוה הקרקטר הראשון חלק מהאינפורמציה הנכתבת ואין לו משמעות מיוחדת.

2) קים המושג של רקורד על הסרט. בקריאה מכרטיסים היה הרקורד - כרטיס. בהדפסה היה הרקורד - שורה. בכתיבה על סרט סימן סוף הרקורד הוא מיגנוט מיוחד והוא נרשם כאשר המחשב מגיע לסוף של פורמט או לקו נטוי בפורמט. אורך רקורד מוגבל בדרך כלל ובמחשב CDC 6400 אינו יכול לעלות על 136 קרקטרים.

דוגמאות לקריאה וכתיבה:

(1

```
     WRITE (1, 1000) (X(I), I = 1, 10)
1000 FØRMAT (5F 15.7/5F 15.7)
```

על כונן מס׳ 1 נכתבים שני רקורדים, כל אחד בן 75 קרקטרים.

(2

```
     READ (3, 500) L, (Y(I), I = 1, L)
500  FØRMAT (I3/20F5.2)
```

מכונן מס׳ 3 נקרא רקורד המכיל את המספר L ואח"כ נקרא רקורד המכיל L מספרים, כל אחד בן 5 קרקטרים.

בזמן כתיבת הרקורדים יש לדאוג לכך שגודלו של L לא יעלה על 20 (ראה הסבר כללי בעמ׳ 56).

REWIND k פרושו לגלגל את הסרט שעל כונן k לתחילתו.

BACKSPACE k פרושו להזיז את הסרט שעל כונן k רקורד אחד בחזרה.

כרטיס PRØGRAM : במחשב CDC 6400 יש לציין מראש את מספרי הכוננים שישתמשו בהם במשך התכנית. דבר זה נעשה ע"י הכנסת כרטיס PRØGRAM שהוא הכרטיס הראשון בתכנית. צורתו הכללית (פרמטרים) שם PRØGRAM הפרמטרים הם מהצורה TAPEi כאשר i מספר כונן בו משתמשת התכנית. השם הוא שם התכנית והוא כפוף לכללים של שמות משתנים.

דוגמא: תכנית בשם TEST המשתמשת בכוננים 1,3 מתחילה בכרטיס PRØGRAM TEST (TAPE1, TAPE3) .

אם התכנית קוראת כרטיסים יש להוסיף את הפרמטר INPUT , אם היא מנקבת כרטיסים יש להוסיף את הפרמטר PUNCH ואם היא מדפיסה תוצאות יש להוסיף גם את הפרמטר ØUTPUT (פרמטר זה רצוי להכניס לכל כרטיס PRØGRAM).

אם אין כרטיס PRØGRAM תבין זאת התכנית כאילו קים הכרטיס PRØGRAM (INPUT, ØUTPUT) , (כלומר תכנית שאין לה שם ואינה משתמשת בסרטים).

מושג ה- File :

File על סרט מגנטי הוא אוסף של אינפורמציה העומד בפני עצמו. גדלו של ה- File אינו מוגבל, הוא יכול להיות קטן מאוד ויכול גם להשתרע על כמה סרטים. אפשר לרשום כמה Files על סרט כך שכל אחד מהם יכיל אינפורמציה מסוג שונה.

התכניתן קובע בהתאם לנסיונו איזה אינפורמציה כדאי להכניס ל- File וכמה File לרשום על הסרט.

בסוף כל File נמצא סימן מיוחד הנקרא end - of - file (או file mark). במחשב CDC 6400 אפשר להבחין בסימן זה בעזרת הפקודה IF(EØF, k) n_1, n_2.

המשמעות - אם פקודת הקריאה האחרונה מכונן k קראה את הסימן של סוף File אז הפקודה -- IF(EØF תגרום לקפיצה אל הפקודה שמספרה n_1 . אחרת תתבצע קפיצה לפקודה שמספרה n_2.

דוגמא:

```
 10 READ (1, 300) A, B, C
300 FØRMAT (3F3. 0)
    IF (EØF, 1) 100, 20
 20 ———
     :
    GØ TØ 10
100 STØP
    END
```

זהו קטע של תכנית הקוראת נתונים מכונן 1. בקריאה רגילה של נתונים עוברת התכנית מהפקודה -- IF(EOF לפקודה 20, שם מעובדים הנתונים ואח"כ מתבצעת קפיצה לפקודה 10 לקריאת הרקורד הבא.

כאשר נקרא end - of - file , עוברת התכנית מהפקודה -- IF(EØF לפקודה 100 ועוצרת שם.

ENDFILE k זו פקודה הרושמת end - of - file על הסרט שבכונן k . בצורה זו סוגרים את ה- file לאחר שנסתימה כתיבתו.

דוגמא:

זהו קטע של תכנית הקוראת
1000 כרטיסי נתונים ומעלה
אותם על סרט בתור file .
בסיום הקטע נסגר ה- file
ע"י end - of - file.

```
           .
           .
           .
    DØ 10 I = 1, 1000
    READ 100, (A (J), J = 1, 8)
100 FØRMAT (8F10, 3)
 10 WRITE(1, 100) (A(J), J = 1, 8)
    ENDFILE 1
           .
           .
           .
```

פרק 12

דוגמאות

שיטת Newton - Raphson למציאת שרש של פונקציה.

נתונה פונקציה $F(X)$. שרש של הפונקציה הוא מספר $X0$ המקום $F(X0) = 0$. דוגמא $F(X) = X^2 - 25$.
שרש של פונקציה זו הוא המספר 5 או המספר 5-.

שיטת ניוטון - רפסון נותנת שרש של הרבה פונקציות, זוהי שיטה פשוטה ולכן קל לתכנת אותה, אולם היא כפופה לכמה הגבלות מתמטיות (שלא יפורטו כאן) ולכן אינה טובה לכל הפונקציות.

השיטה היא איטרטיבית כלומר חוזרים בה על אותו תהליך כמה פעמים, ובכל פעם מתקרבים יותר לפתרון המבוקש. במלים אחרות, אם X_i הוא קירוב מסוים לפתרון, אז קיים תהליך הנותן מספר X_{i+1} שהוא קירוב יותר טוב מ - X_i .

התהליך הוא $X_{i+1} = X_i - \frac{F(X_i)}{F'(X_i)}$.

דוגמא: $F(X) = X^2 - 25$

כאן $F'(X) = 2X$ ולכן תהליך ניוטון - רפסון יהיה

$$X_{i+1} = X_i - \frac{X_i^2 - 25}{2X_i}$$

בתור קירוב ראשון נבחר באופן מקרי את המספר 2. נסמן $X_1=2$
ואז הקירוב הבא X_2 ינתן ע"י

$$X_2 = X_1 - \frac{X_1^2 - 25}{2X_1} = 2 - \frac{4 - 25}{2 \cdot 2} = 7.25$$

הקירוב הבא X_3 יהיה

$$X_3 = X_2 - \frac{X_2^2 - 25}{2\,X_2} = 7.25 - \frac{7.25^2 - 25}{2 \cdot 7.25} = 5.35$$

שתי החזרות (איטרציות) הבאות תיתנה

$X_4 = 5.0114$ $\qquad$ $X_5 = 5.00001$

כך מתקרבים למספר 5 שהוא כאמור, שרש של $F(X)$.

דוגמא: כאן $F(X) = aX^2 + bX + c$

ולכן יהיה תהליך $F'(X) = 2aX + b$

ניוטון - רפסון

$$X_{i+1} = X_i - \frac{aX^2 + bX + c}{2aX + b}$$

תהליך זה ימצא שרש של $F(X)$ כלומר אחד מהפתרונות של המשואה הרבועית $aX^2 + bX + c = 0$.

דוגמא: $F(X) = X^2 - a$ כאשר a - מספר חיובי נתון. שרש של פונקציה זו יקיים $X^2 - a = 0$ או $X = \pm\sqrt{a}$ ולכן נותנת שיטת ניוטון - רפסון אפשרות נוחה לחישוב שרש.

תהליך האיטרציה יהיה כאן

$$X_{i+1} = X_i - \frac{X_i^2 - a}{2\,X_i} = \frac{1}{2}\left(X_i + \frac{a}{X_i}\right)$$

כדי להתחיל תהליך איטרציה של ניוטון - רפסון יש לבחור קירוב ראשון X_1 כלשהו. X_1 יכול להיות למשל המספר 1, רצוי לבחור בתור קירוב ראשון, מספר שיודעים עליו שהוא מאותו סדר גודל של הפתרון, אולם אם אי אפשר לעשות זאת, הרי שאפשר לבחור קירוב ראשון כלשהו.

בזמן בצוע האיטרציות מתקרבים לפתרון אולם עלול לקרות שהפתרון המדויק לא יושג אף פעם. עלול לקרות ש-$F(X_i)$

קרב לאפס אולם לא יתאפס. במקרה כזה יש לדאוג לכך שהעבודה
פסק כאשר הפתרון המקורב הוא מספיק מדויק.

במלים אחרות יש להפסיק כאשר $F(X_i)$ מספיק קרוב לאפס.

שיטה פשוטה הנותנת זאת היא חישוב $F(X_{i+1})$ לאחר
איטרציה ובדיקה האם $F(X_{i+1})$ 'מספיק' קרוב לאפס.

שיטה זו מחיבת חישוב של הפונקציה F לאחר כל איטרציה,
הסרון ניכר. חישוב הפונקציה F עלול להיות מסובך
גזול זמן מחשב.

שיטה שניה מתבססת על העובדה שככל שמתקרבים לפתרון
דויק, הולכת מהירות ההתקרבות וקטנה. במלים אחרות, ההפרש
ן שני קירובים עוקבים הולך וקטן.

לאחר כל איטרציה מחשבים את ההפרש $|X_{i+1} - X_i|$
פסיקים את העבודה כאשר ערכו של הפרש זה מספיק קטן (נאמר
ן מ- 10^{-4}).

שיטת ניוטון - רפסון אינה מובילה תמיד לפתרון. קיימות
נקציות לגביהן אפשר לבצע את התהליך מבלי להתקרב לפתרון.
מקרים כאלה יהיה ערכו של ההפרש $|X_{i+1} - X_i|$ תמיד
דול מאפס, ויש להפסיק את העבודה לאחר מספר מסוים (קבוע
ראש) של חזרות.

גם במידה ומוצאים שרש של הפונקציה, יתכן שיש לה עוד
רשים. שיטת ניוטון - רפסון אינה אומרת כלום לגבי שאר השרשים
ילם לפעמים אפשר למצוא שרש אחר ע"י בחירת קירוב התחלתי
X_1 אחר.

התכנית הבאה היא תכנית כללית לחישוב שרש של פונקציה כלשהי
$F(X)$. התכנית מתחילה מקירוב התחלתי 1 ומסיימת

כאשר ההפרש $|X_{i+1} - X_i|$ קטן מ- 10^{-6}. אם תוך 1000 חזרות לא קטן ההפרש מ- 10^{-6}, נפסקת העבודה והתכנית מדפיסה את המלים NØ CØNVERGENCE.

```
      PRØGRAM NR(INPUT, ØUTPUT)
      X0 = 1.
      DØ 10 I = 1, 1000
      X1 = X0 - F (X0) / FT (X0)
      IF (ABS(X1 -X0) . LE.  1.E - 6) GØ TØ 20
   10 X0 = X1
      PRINT 1000
 1000 FØRMAT (15 H NØ CØNVERGENCE)
      STØP
   20 Y = F (X1)
      PRINT 1100, X1, Y
 1100 FØRMAT (1X, E15.8, F9.5)
      STØP
      END
      FUNCTIØN F (X)
      .
      .
      END
      FUNCTIØN FT (X)
      .
      .
      END
```

לתכנית מצורפות שתי פונקציות F, FT

F מחשבת את הפונקציה

FT מחשבת את הנגזרת שלה.

פונקציות אלה יכולות להיות כלשהן, כלומר יתכן שיהיו $F = X^2 - 25$ $FT = 2X$ ואז התכנית תחשב שרש רבועי של 25.

יתכן שיהיו $F = aX^2 + bX + c$ $FT = 2aX + b$ ואז התכנית תחשב שרש של המשואה הרבועית $aX^2 + bX + c = 0$.

בעית שמונה המלכות

נתון לוח שחמט ועליו שמונה מלכות. הבעיה היא לסדר את המלכות על הלוח באופן שאף מלכה לא תאיים על אף מלכה אחרת.

הסבר: כל מלכה מאיימת על השורה, העמוד והאלכסונים (או האלכסון) בהם היא נמצאת. כדי שאף מלכה לא תאיים על האחרות, היא חייבת לעמוד לבד בשורה, בעמוד ובאלכסונים (אלכסון) שלה.

מאחר ולוח שחמט מכיל שמונה שורות וכל מלכה חייבת לעמוד לבד בשורתה, יוצא שכל שורה חייבת להכיל בדיוק מלכה אחת.

מאותם שקולים נובע שכל עמוד חייב להכיל בדיוק מלכה אחת.

תכנית המנסה לפתור את הבעיה יכולה לכן לעבור על פני לוח השחמט שורה שורה. בכל שורה תציב התכנית מלכה ותדאג לכך שהיא תוצב בעמוד ריק. נוסף לכך יש לדאוג שלא יהיה איום על המלכה דרך האלכסונים בהם היא עומדת.

התכנית המודגמת כאן מכילה וקטור בשם TUR בעל שמונה מקומות. במקום ה- K בוקטור זה נמצא מספרו של העמוד עליו ניצבת המלכה הנמצאת בשורה ה- K.

וקטור בשם MAK מכיל במקום ה- K שלו, 1 - אם העמוד ה- K תפוס, 0 - אם הוא פנוי.

בצורה זו יכולה התכנית להציב כל מלכה בעמוד פנוי.

לגבי האלכסונים. מלכה הניצבת על הלוח בשורה ה- I ובעמוד ה- J , ניצבת על שני אלכסונים. אחד מהם הוא אוסף כל המשבצות לגביהן הסכום I + J קבוע, האחר הוא אוסף כל המשבצות לגביהן קבוע ההפרש J - I .

התכנית מכילה שני וקטורים MINUS (8) ,PLUS(8). כאשר מוצבת מלכה בשורה ה- I בעמוד ה- J מוכנס הסכום I + J למקום PLUS (I) וההפרש J - I למקום MINUS (I) . בצורה זו יודעת התכנית באיזה אלכסונים נמצאת המלכה שבשורה ה- I .

התכנית מתחילה מהערך I = 1 מכניסה מלכה לשורה ה- I , בודקת מהו העמוד הראשון הפנוי בשורה זו ובודקת אם שני האלכסונים המתאימים פנויים. אם נמצא מקום כזה בשורה ה- I , מכניסה התכנית את המלכה למקום זה ומסמנת את העמוד המתאים ואת האלכסונים המתאימים כתפוסים. לאחר מכן מוגדל I באחד והתהליך חוזר.

אם בשורה ה- I לא נמצא מקום פנוי, חוזרת התכנית לשורה ה- I - 1 , מזיזה את המלכה הנמצאת שם למקום פנוי אחר ושוב חוזרת לשורה ה- I .

לאחר בצוע התהליך על שמונה השורות מודפס הוקטור TUR וכן מודפס לוח השחמט עם המלכות במקומות המתאימים.

```
                   PROGRAM QQ(OUTPUT)
000003             INTEGER TUR(8),MAK(8),PLUS(8),MINUS(8),II(8)
000003             DO 10 I=1,8
000005          10 TUR(I)=MAK(I)=PLUS(I)=MINUS(I)=0
000014             I=1
000015          30 IF(I.EQ.9) GO TO 100
000017          40 J=TUR(I)
000021             IF(J.EQ.0) GO TO 50
000022             MAK(J)=0
000023          50 J=J+1
000025             IF(J.EQ.9) GO TO 80
000027             IF(MAK(J).NE.0) GO TO 50
000030             JPI=J+I
000031             JMI=J-I
000031             IM=I-1
000032             IF(IM.EQ.0) GO TO 70
000034             DO 60 K=1,IM
000035             IF(JPI.EQ.PLUS(K).OR.JMI.EQ.MINUS(K)) GO TO 50
000044          60 CONTINUE
000046          70 TUR(I)=J
000050             MAK(J)=1
000051             PLUS(I)=J+I
000052             MINUS(I)=J-I
000053             I=I+1
000055             GO TO 30
000055          80 TUR(I)=0
000057             I=I-1
000060             GO TO 40
000060         100 PRINT 1000,TUR
000066        1000 FORMAT (1H1,8I2//)
000066             DO 120 I=1,8,2
000070             II(1)=II(3)=II(5)=II(7)=(1H )
000076             II(2)=II(4)=II(6)=II(8)=(1H+)
000103             K=TUR(I)
000105             II(K)=(1HQ)
000107             PRINT 1100,II
000115        1100 FORMAT (1X,33(1H+)/2H +,4(1X,A1,1X,2H++,A1,2H++
000115             II(1)=II(3)=II(5)=II(7)=(1H+)
000123             II(2)=II(4)=II(6)=II(8)=(1H )
000130             K=TUR(I+1)
000132             II(K)=(1HQ)
000134         120 PRINT 1200,II
000144        1200 FORMAT (1X,33(1H+)/2H +,4(1H+,A1,2H++,1X,A1,2H
000144             PRINT 1300
000150        1300 FORMAT (1X,33(1H+))
000150             STOP
000152             END
```

1 5 8 6 3 7 2 4

```
+++++++++++++++++++++++++++++++++
+ Q +++++   +++++   +++++   +++++
+++++++++++++++++++++++++++++++++
+++++   +++++   ++Q++   +++++   +
+++++++++++++++++++++++++++++++++
+   +++++   +++++   +++++   ++Q++
+++++++++++++++++++++++++++++++++
+++++   +++++   +++++ Q +++++   +
+++++++++++++++++++++++++++++++++
+   +++++ Q +++++   +++++   +++++
+++++++++++++++++++++++++++++++++
+++++   +++++   +++++   ++Q++   +
+++++++++++++++++++++++++++++++++
+   ++Q++   +++++   +++++   +++++
+++++++++++++++++++++++++++++++++
+++++   +++++ Q +++++   +++++   +
+++++++++++++++++++++++++++++++++
```

חישובי ממוצעים, סטיות תקן ומקדמי מיתאם

התכנית הבאה קוראת חמשה טורים של מספרים, מחשבת את הממוצע ואת סטית התקן של כל טור ומחשבת את מקדמי המיתאם של כל טור עם כל הטורים האחרים.

הממוצע $\bar{X}$ של טור המספרים Xi ניתן ע"י הנוסחה $\bar{X} = \frac{1}{n}\sum Xi$. סטית התקן σ_x ניתנת ע"י הנוסחה $\sigma_x = \sqrt{\frac{1}{n}\sum Xi^2 - (\bar{X})^2}$ מקדם המיתאם Rxy בין שני טורי המספרים Xi Yi ניתן ע"י הנוסחה

$$Rxy = \frac{\frac{1}{n}\sum XiYi - \bar{X}\bar{Y}}{\sigma_x \sigma_y}$$

התכנית קוראת כרטיס ובו הערך של n (ארכו של כל טור מספרים). לאחר מכן היא קוראת n כרטיסים שבכל אחד מהם 5 מספרים. מספרים אלה מתוספים לוקטור X, ורבועיהם מתוספים לוקטור XS לאחר קריאת הכרטיסים מכיל כל אלמנט בוקטור X את סכום המספרים של אחד הטורים, וכל אלמנט בוקטור XS, את סכום רבועי המספרים של אותו טור. נוסף לכך מתוספות המכפלות המעורבות של המספרים מכל כרטיס, למטריצה R.

בשלב הבא מחשבת התכנית את הממוצעים בוקטור X, את סטיות התקן בוקטור XS ואת מקדמי המיתאם במטריצה R. הוקטורים X, XS מודפסים ולבסוף מודפסת המחצית השמאלית התחתונה של המטריצה R (שהיא סימטרית ולכן אין צורך להדפיס את כולה).

```
                PROGRAM MSUR(INPUT,OUTPUT)
000003          DIMENSION T(5),X(5),XS(5),R(5,5)
000003          PRINT 900
000007      900 FORMAT (1H1)
000007          DO 10 I=1,5
000011          T(I)=X(I)=XS(I)=0
000015          DO 10 J=1,5
000016       10 R(I,J)=0
000025          READ 1000,N
000032     1000 FORMAT (I3)
000032          DO 30 I=1,N
000034          READ 1100,T
000041     1100 FORMAT (5F5.2)
000041          PRINT 1200,T
000047     1200 FORMAT (6X,5F7.2)
000047          DO 20 J=1,5
000051          X(J)=X(J)+T(J)
000053          XS(J)=XS(J)+T(J)*T(J)
000056          DO 20 K=1,5
000060       20 R(J,K)=R(J,K)+T(J)*T(K)
000073       30 CONTINUE
000075          DO 40 I=1,5
000077          X(I)=X(I)/N
000102       40 XS(I)=SQRT(XS(I)/N-X(I)*X(I))
000113          DO 50 I=1,5
000115          DO 50 J=1,5
000115       50 R(I,J)=(R(I,J)/N-X(I)*X(J))/(XS(I)*XS(J))
000140          PRINT 1300,X,XS
000147     1300 FORMAT (6H MEANS,5F7.2/6H S. D.,5F7.2///18X,11HCORR. COEF./)
000147          PRINT 1400,(I,I=1,5)
000160     1400 FORMAT (4X,5I7/)
000160          DO 60 I=1,5
000162       60 PRINT 1500,I,(R(I,J),J=1,I)
000201     1500 FORMAT (1X,I2,3X,5F7.2)
000201          STOP
000203          END
```

	.01	.02	.03	.04	.05
	.10	.20	.30	.40	.50
	5.00	4.00	3.00	2.00	1.00
	123.45	543.21	534.21	231.54	134.25
	0.00	4.75	3.57	86.50	.12
	45.60	4.86	85.20	125.98	5.69
	42.15	2.38	.02	3.58	2.20
	65.32	7.85	.63	.01	24.75
	.27	3.05	.69	.02	12.35
	584.32	52.03	6.52	.85	.03
MEANS	86.62	62.23	63.42	45.09	18.09
S. D.	170.23	160.99	158.90	75.27	39.44

CORR. OOEF.

	1	2	3	4	5
1	1.00				
2	.16	1.00			
3	.07	.98	1.00		
4	-.03	.82	.89	1.00	
5	.04	.98	.97	.80	1.00

חפוש בטבלה.

נתונה טבלת מספרים ויש למצוא האם מספר A נתון מופיע בה ואם כן איפה.

השיטה הפשוטה היא לעבור על כל הטבלה מספר אחרי מספר. השיטה נקראת חפוש לינארי והיא טובה לטבלאות קטנות. בטבלאות גדולות עלול חפוש כזה להיות בזבזני בזמן מחשב.

שיטה מהירה יותר, הטובה לטבלאות גדולות היא שיטת החפוש הבינרי. שיטה זו טובה רק לטבלאות הממוינות בסדר עולה ועקרונה הוא כדלקמן. משוים את המספר A למספר הנמצא באמצע הטבלה. אם A גדול ממספר זה אזי עוברים למחצית העליונה של הטבלה. אם A קטן מהמספר האמצעי אזי עוברים למחצית התחתונה של הטבלה. אותו תהליך חוזר על עצמו לגבי המחצית שנבחרה. גם שם בוחרים מספר אמצעי, משוים אליו את A ובוחרים רבע עליון או רבע תחתון. כך מחלקים את הטבלה המקורית לחלקים הולכים וקטנים עד שמגיעים לחלק המכיל מספר יחיד, אם מספר זה שוה ל- A הרי שהחפוש נגמר, אחרת A אינו בטבלה.

שיטה זו מהירה בהרבה מהחפוש הלינארי. חסרונה הוא שדרוש מיון מוקדם של המספרים בטבלה.

הסוברוטינה שבדוגמא מקבלת טבלה X באורך N ומספר KEY שאותו יש לחפש. התוצאה מוכנסת למשתנה MID . אם המספר שב- KEY אינו נמצא בטבלה, מכניסה הסוברוטינה מספר שלילי לתוך MID וחוזרת.

המשתנים IGH, LØW מכילים את הגבולות העליון והתחתון של חלק הטבלה שבו מחפשים בכל שלב.

הדפסת התוצאות מכילה שני חלקים. בתחילת כל חלק מודפס ערכו של N (11 בדוגמא) והמספרים שבטבלה.

בחלק הראשון מחפשת התכנית את המספרים 11 - -,1,2 בטבלה בשיטה הלינארית. בחלק השני מחפשת התכנית את אותם המספרים בשיטה הבינרית וזאת לאחר מיון הטבלה. המספרים 6,7,8,10,11 אינם מופיעים בטבלה ולכן מודפסים לידם ערכים שליליים.

```
                 REAL A(25)
000033           READ 1000,N,(A(I),I=1,N)
000020     1000  FORMAT (I2,25F3.0)
000020           PRINT 1100,N,(A(I),I=1,N)
000035     1100  FORMAT (1X,I3,25F4.0)
           C SEARCH THE NUMBERS 1,2,...,N IN LINEAR MODE
000035           DO 10 I=1,N
000037           F=I
000040           CALL LINSRCH(F,A,N,K)
000043       10  PRINT 1200,F,K
000056     1200  FORMAT (1X,F5.0,I3)
           C --- SORT THE TABLE A ---
000056           N1=N-1
000060           DO 20 I=1,N1
000061           I1=I+1
000063           DO 20 J=I1,N
000065           IF(A(I).LE.A(J)) GO TO 20
000070           F=A(I)
000071           A(I)=A(J)
000074           A(J)=F
000075       20  CONTINUE
000102           PRINT 1100,N,(A(I),I=1,N)
           C SEARCH THE NUMBERS 1,2,...,N IN RINARY MODE
000117           DO 30 I=1,N
000121           F=I
000122           CALL BINSRCH(F,A,N,K)
000125       30  PRINT 1200,F,K
000140           STOP
000142           END
002310


                 SUBROUTINE LINSRCH(KEY,X,N,IIM)
000007           REAL X(N),KEY
000007           DO 10 IIM=1,N
000010           IF(X(IIM).EQ.KEY) RETURN
000012       10  CONTINUE
000014           IIM=-1
000015           RETURN
000016           END
002600


                 SUBROUTINE BINSRCH(KEY,X,N,MID)
000007           REAL X(N),KEY
000007           IGH=N
000007           LOW=1
000011       10  MID=(IGH+LOW)/2
000013           IF(KEY-X(MID)) 30,20,40
000017       20  RETURN
000020       30  IGH=MID-1
000022           GO TO 50
000022       40  LOW=MID+1
000024       50  IF(IGH.GE.LOW) GO TO 10
000027           MID=-MID
000030           RETURN
000030           END
002600
```

```
 11   2  -4   5   9   3   4   1  15  34   9   0
    1  7
    2  1
    3  5
    4  6
    5  3
    6 -1
    7 -1
    8 -1
    9  4
   10 -1
   11 -1
 11  -4   0   1   2   3   4   5   9   9  15  34
    1  3
    2  4
    3  5
    4  6
    5  7
    6 -8
    7 -8
    8 -8
    9  9
   10-10
   11-10
```

נספח מס׳ 1

רשימת הפונקציות הבנויות

הערה: רשימה זו כוללת את הפונקציות הבנויות הקיימות ברוב המחשבים. עם זאת אפשר למצוא בכל מחשב פונקציות נוספות שאינן כלולות ברשימה זו. הסברים על פונקציות אלה אפשר למצוא בספרי העזר של כל מחשב.

שם	תאור	פרמטר	ערך
ABS (X)	ערך מוחלט	מ	מ
AMAX1(X1, X2, . . .)	הערך של הפרמטר המקסימלי	מ	מ
AMIN1(X1, X2, . . .)	הערך של הפרמטר המינימלי	מ	מ
FLØAT (I)	הפיכת הפרמטר לממשי	ש	מ
IABS (I)	ערך מוחלט	ש	ש
INT (X)	הפיכת הפרמטר לשלם תוך כדי קיצוצו	מ	ש
ISIGN (I1, I2)	הסימן של I 1 כפול בערך המוחלט של I 2	ש	ש
SIGN (X1, X2)	הסימן של X1 כפול בערך המוחלט של X2	מ	מ
MØD (I1, I2)	הערך של I 1 כשהוא מחושב מודולו I 2	ש	ש
ACØS (X)	arccosine ברדיאנים	מ	מ
ALØG (X)	לוגריתמוס טבעי	מ	מ

שם	תאור	פרמטר	ערך
ALØG 10(X)	לוגריתמוס על בסיס 10	מ	מ
ASIN (X)	arcsine ברדיאנים	מ	מ
ATAN (X)	arctg ברדיאנים	מ	מ
EXP (X)	e^x	מ	מ
SIN (X)	סינוס ברדיאנים	מ	מ
SQRT (X)	$\sqrt{X}$	מ	מ
TAN (X)	tg x	מ	מ
TANH (X)	טנגנס היפרבולי ברדיאנים	מ	מ

נספח מס׳ 2

המשפטים בשפת FORTRAN

1) משפט יחוס אריתמטי (עמוד 12) צורתו $a = e$ כאשר a שם של משתנה , e - בטוי אריתמטי. הבטוי האריתמטי e מחושב, ותוצאת החישוב מועברת למשתנה a .

2) " GØ TØ ללא תנאי" (עמוד 16) צורתו GØ TØ n כאשר n היא מס׳ זהוי של פקודה.
המחשב יקפוץ לפקודה שמספרה n , יבצע אותה וימשיך ממנה.

3) " GØ TØ מחושב" (עמוד 16) צורתו
GØ TØ $(n_1, n_2, \ldots, n_k), i$ הם מספרי זהוי של n_j
פקודות, i הוא שם של משתנה שלם. אם i מכיל את המספר
j , יקפוץ המחשב לפקודה שמספרה n_j , יבצע אותה וימשיך ממנה.

4) STØP (עמוד 18).

5) END (עמוד 19).

6) " IF - אריתמטי" (עמוד 19).

צורתו IF $(e)n_1, n_2, n_3$

e היא בטוי אריתמטי, n_j הם מספרי זהוי של פקודות.
המחשב יקפוץ לפקודה שמספרה n_1 אם ערכו של e הוא שלילי,
לפקודה שמספרה n_2 אם $e = 0$ ולפקודה שמספרה n_3 אם
e חיובי.

7) " IF - לוגי" (עמוד 22) צורתו IF(l)s , כאשר l הוא בטוי לוגי, s הוא משפט שיכול להיות משפט כלשהו פרט ל " IF לוגי" ופרט ל " DØ ".
אם הבטוי הלוגי l הוא "נכון", יבוצע המשפט s . אם l "אינו נכון", יעבור המחשב לפקודה הבאה מבלי לבצע את s .

8) DIMENSIØN (עמוד 24) צורתה הכללית

DIMENSIØN m_1(ממד), m_2 (ממד), . . .

9) INTEGER REAL (עמוד 25) מקנות מעמד של שלם או ממשי למשתנים מסוימים.

10) " DØ " (עמוד 29)

צורתה הכללית $DØ\ n\ i = m_1, m_2, m_3$

או $DØ\ n\ i = m_1, m_2$

המחשב יבצע את שורת הפקודות החל מהפקודה שאחרי ה" DØ " ועד הפקודה שמספרה n (כולל אותה), כאשר המשתנה i גדל מ- m_1 ל- m_2 בקפיצות של m_3 .

11) " READ PRINT " (עמוד 39)

צורתן הכללית READ n, list

PRINT n, list

n הוא מספר פקודת פורמט. list הוא רשימה של שמות משתנים.

12) פורמט (עמוד 42).

הצורה הכללית n FØRMAT(שדה, שדה, ...)

n הוא מספר זהוי החיב להופיע.

13) " CALL " (עמוד 61)

צורתה הכללית CALL שם (פרמטר, פרמטר, ...)

או CALL שם

המחשב יקפוץ לסוברוטינה ששמה צוין בפקודה, ימסור לה את ערכי הפרמטרים המתאימים ולאחר שזו תגמור את פעולותיה, יחזור המחשב לתכנית הראשית, לפקודה שאחרי ה- CALL.

14) " RETURN " (עמוד 63)

בעזרת פקודה זו חוזרים מתכנית משנה לתכנית הראשית.

15) " CØMMØN " (עמוד 71)

הצורה הכללית

CØMMØN list

המשתנים המופיעים ב- list יהיו משותפים לתכנית הראשית ולכל

תכנית משנה בה יופיע ה- CØMMØN

16) משפט יחוס אלפמרי (עמוד 82)

צורתו הכללית

$a = (nHa_1a_2...a_n)$

אגף ימין הוא קבוע אלפמרי והוא נכנס למשתנה שבאגף שמאל.

17) DATA (עמוד 83)

צורתו הכללית

DATA list/clist/

list היא רשימה של משתנים.

clist היא רשימה של קבועים.

פקודה זו מכניסה את הקבועים לתוך המשתנים ועושה זאת רק פעם

אחת בתחילת התכנית.

פתרונות לתרגילים

עמוד 12:

$(x + y)^3$ → (X + Y) ** 3

$a + \frac{b}{c + d^2}$ → A + B/ (C + D**2)

$\sqrt{j}$ → J**.5 אפשרי רק כאשר J ממשי

$1/(1 - \frac{v^2}{c^2})\ f$ → 1/ (1 - V**2/ C**2) / F

עמוד 15:

	הערך הנכנס לאגף שמאל	ערכו של אגף ימין
A	3.	3.
I	5	5
A	5.	5.
A	1.	1.
A	5.	5
I	0	0

עמוד 23:

```
read X
IF (X. LT. 0. ØR. X. GE. 50. ) STØP
IF (X. GE. 20. )GØ TØ 10
Y = -1.
GØ TØ 30
```

```
10  IF (X.GE.40.)GØ TØ 20
    Y = 0
    GØ TØ 30
20  Y = 1.
30  print  Y
    STØP
    END
```

:עמוד 38

(1

```
    DIMENSIØN  Y(100),  W(100)
    S = 0
    DØ 10 I = 1, 100
10  S = S + Y(I) *W(I)
    print S
    STØP
    END
```

(2

```
    DIMENSIØN  A(3, 7),  B(3, 7)
    DØ 10 I = 1, 3
    DØ 10 J = 1, 7
10  A(I, J) = A(I, J) + B(I, J)
    print  A
    STØP
    END
```

(3

```
      DIMENSIØN  A(5, 6), B(6, 8), C(5, 8)
      DØ 3 I = 1, 5
      DØ 3 J = 1, 8
      C (I, J) = 0
      DØ 3 K = 1, 6
    3 C(I, J) = C(I, J) + A (I, K)*B (K, J)
      print  C
      STØP
      END
```

<u>עמוד 90:</u>

(1

```
      DIMENSIØN  L(104), LIN(104)
      J = 1
      DØ 10 I = 1, 629, 6
      X = I - 1
      X = X/100.
      L (J) = 26.5 - SIN (X)*25.
   10 J = J + 1
      DØ 30 I = 1, 51
      DØ 20 J = 1, 104
      LIN (J) = (1Hb)
      IF(I.EQ.26) LIN (J) = (1H-)
   20 IF (L(J).EQ.I) LIN (J) = (1H* )
   30 PRINT  1000, LIN
 1000 FØRMAT (1X,  104A1)
      STØP
      END
```

תכנית זו מצירת את הגרף על פני 51 שורות. בכל שורה מודפסים 104 קרקטרים ולכן המרחק בין כל שני קרקטרים חיב להתאים לשש מאיות רדיאן (כי $2\pi \approx 6.24 = 0.06 \cdot 104$).

בחלק הראשון של התכנית מקבל המשתנה x את הערכים מאפס עד 6.24 בקפיצות של 06.

sin x עובר טרנספורמציה לינארית ההופכת את ערכי הסינוס (בין 1 - ל 1 +) למספרים שלמים בין 51 ל 1 . מספרים אלה מוכנסים לוקטור L וכעת מכיל האלמנט L(J) את מספר השורה (בין 1 ל 51) בה חיב להיות כוכב במקום ה- J .

בחלק השני של התכנית מודפסות 51 שורות. לפני הדפסת כל שורה עוברת התכנית על הוקטור L ובודקת היכן יש להכניס כוכבים לשורה זו.

(2

```
      DIMENSIØN LIN(131)
      N = 50
      R = N/12.
      R = R*R
      DØ 10 I = 1, 131
10    LIN (I) = (1Hb)
      DØ 20 I = 1, N
      Y = I - N/2
      Y = Y/6.
      X = SQRT (R - Y*Y) *10.
      J1 = 66.5 + X
      J2 = 66.5 - X
      LIN (J1) = LIN (J2) = (1H*)
```

```
      PRINT  1000, (LIN(J), J = 1, 131)
1000  FØRMAT (1X, 131A1)
  20  LIN (J1) = LIN (J2) = (1Hb)
      STØP
      END
```

4.80

הדפיס אקדמון
טל 36253-4 ירושלים